DeployDirector

Version 2.6

Administrator's Guide

World Headquarters
8001 Irvine Center Drive
Irvine, CA 92618
www.quest.com
email: info@quest.com

April 2003

STDDAG26

Warranty

The information contained in this document is subject to change without notice. Quest Software makes no warranty of any kind with respect to this information. QUEST SOFTWARE SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTY OF THE MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Quest Software shall not be liable for any direct, indirect, incidental, consequential, or other damage alleged in connection with the furnishing or use of this information.

Trademarks

DeployDirector™ is a trademark Quest Software, Inc. Other trademarks and registered trademarks used in this guide are property of their respective owners.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

This product contains software developed by Jason Hunter (jhunter@servlets.com), copyright © 2001 Jason Hunter. All rights reserved.

> Redistribution of the com.oreilly.servlet package is permitted provided that the following conditions are met:
>
> 1. You redistribute the package in object code form only (as Java .class files or a .jar file containing the .class files) and only as part of a product that uses the classes as part of its primary functionality.
>
> 2. You reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
>
> THIS SOFTWARE IS PROVIDED BY THE AUTHOR “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Quest Software

World Headquarters
8001 Irvine Center Drive
Irvine, CA 92618
www.quest.com
e-mail: info@quest.com
U.S. and Canada: 949.754.8000

Please refer to our Web site for regional and international office information.

About Quest

Quest Software, Inc. (NASDAQ: QSFT) is a leading provider of application management solutions. Quest provides customers with Application Confidence[sm] by delivering reliable software products to develop, deploy, manage and maintain enterprise applications without expensive downtime or business interruption. Targeting high availability, monitoring, database management and Microsoft infrastructure management, Quest products increase the performance and uptime of business-critical applications and enable IT professionals to achieve more with fewer resources. Headquartered in Irvine, Calif., Quest Software has offices around the globe and more than 18,000 global customers, including 75% of the Fortune 500. For more information on Quest Software, visit www.quest.com.

Contacting Quest Software

Phone	949.754.8000 (United States and Canada)
E-mail	info@quest.com
Mail	Quest Software, Inc. World Headquarters 8001 Irvine Center Drive Irvine, CA 92618 USA
Web site	www.quest.com

Please refer to our Web site for regional and international office information.

Contents

Chapter 5 Configuring Bundle Installation Properties

Chapter 6 Configuring Bundle Runtime Properties

Chapter 9 End User and Administrator Access

Chapter 10 Viewing and Managing Logs

Chapter 11 Customizing Functionality with the SDK

Chapter 1
Installation and Setup

This introductory chapter contains information on system requirements, product installation, license setup and Administration Tool deployment. For current release information (including new features and known problems), please refer to the readme.

Important: If you are upgrading your version of DeployDirector, please first refer to Upgrading the DeployDirector Server on page 24 to ensure you have properly migrated your DeployDirector data, and have prepared your system for a new installation.

Supported Platforms and General Requirements

DeployDirector requires the installation of components on both the client and server sides, which includes components for the server, administrator workstations, and regular clients to which applications will be deployed.

The following sections list supported platforms for these three installation destinations, and outline issues that should be considered when allocating hardware resources for all areas your deployment network.

Server-Side Requirements and Hardware Considerations

While DeployDirector supports a variety of servers and Web server environments, it is important to consider how aspects of your organization's hardware and network infrastructure can affect overall performance of the deployment system. Since each organization's hardware resources and implementation of DeployDirector will vary, specific considerations are list, for which general guidelines are offered.

Tested Platforms:	Tested Application Servers:
HP-UX 11	BEA WebLogic 6.1, 7.0, 8.1
IBM AIX 5.1	IBM WebSphere 5.0
RedHat Linux 7.2, 8.0	Apache Tomcat 4.1.18
Solaris 2.8, 2.9	SunONE 7.0
Windows 2000	
Windows NT 4.0 (SP6)	

Note: DeployDirector is *not* tested with plugin Web servers. However, DeployDirector can be expected to work if the Web and application server combination is already working properly.

Java requirements: DeployDirector's server-side components have been implemented entirely in Java (JDK 1.2 or greater). As such, these components should work on any fully Java-compliant platform.

Storage requirements: Storage requirements are dependent on the number and size of JREs and bundle versions managed by DeployDirector, as well as the amount of work space allocated to some functions. The following items or settings affect server-side storage requirements:

- The core DeployDirector server files require ~37MB of storage space.
- The uncompressed JREs, which are included with the DeployDirector installation, require ~53MB of storage space. Additional space will obviously be needed for other (uncompressed) JREs that your administrators will add after initial installation.

- The number and size of bundles and bundle versions (in an uncompressed form) that your organization plans to manage with DeployDirector.
- The amount of cache space allocated to DeployDirector functions.
- The amount of storage space allocated to logs.

Memory requirements: While it is difficult to state the memory requirements of the DeployDirector server components, the base requirement is dependent on the JDK used to run them. This base amount increases with each bundle that is added to the vault, and that increase is dependent on the content of the bundle that is added.

CPU requirements Some of the factors that affect CPU usage include:

- bundle content,
- compression ratios,
- caching limitations,
- the number of queries made to the server.

Despite the variety of factors, as a rule, a ratio 1 CPU for every 500 clients should suffice in most cases.

Network configuration: The network environment in which DeployDirector is installed has a significant effect on its server-side performance.

For example, if a firewall or proxy exists between the server and clients, network throughput will be slowest at either the server or the firewall. As another example, when clustering is used, and the client load is evenly distributed to the servers in the cluster, the actual network throughput requirements of any one server is the total throughput requirement divided by the number of servers in the cluster.

It is recommended that your organization's network administrators assess existing configurations, and determines what modifications may better accommodate a deployment system.

Network throughput: This factor has the most significant impact on server performance. There are several issues that should be considered:

- the number of clients that exist, and will exist in the future,
- the size of the initial client download,
- the size and frequency of updates,
- whether or not installations and updates will be staggered,
- whether clients will download JREs at the time of bundle installation, or if pre-installed JREs will be used,
- how often clients may be contacting the server,
- the connection / data throughput of the Web or application server being used.

Administrator Client Requirements

The Administrator client is the workstation on which the DeployDirector Administration Tool is installed.

Tested Platforms:	Hardware and JRE Requirements
Windows 2000	64MB RAM (see note below)
Windows NT Workstation 4.0 (SP6)	30MB hard drive space (plus free hard drive space for bundle construction or DAR exporting)
Windows XP	
	JRE 1.4.1 only

Memory Requirements: The recommended amount listed above should suffice to manage the base footprint of the Administration Tool (~26MB), the `version.xml` file of each bundle that is expanded in the Administration Tool's Bundle tab, and the creation of bundles and/or DARs.

Note: The amount of memory required for the last two tasks is proportional to the size and complexity of the bundle that is expanded or being modified.

Client-Side Platform Support and Requirements

Tested Platforms:	Supported Browsers
IBM AIX 5.1	Internet Explorer 5.0 or greater
RedHat Linux 7.3, 8.0	Netscape Navigator 4.7 or greater
Solaris 2.8, 2.9	
Windows 2000	**Supported Desktop Environments:**
Windows 98 SE	Windows, KDE, Gnome, CDE
Windows NT Workstation 4.0 (SP6)	
Windows XP	**Client Requirements:**
Mac OS X	DeployDirector's client-side JDK 1.2 components have been tested with and greater.
HPUX 11	

Storage Requirements: When a bundle is being deployed to a client machine, the total storage required for that session includes the size of the following:

- the uncompressed bundle,
- the DDCAM (~600KB),
- any JRE that is accompanying the bundle (if at all).

The most important consideration for the amount of storage space available on the client side is the size of the bundles that you plan on deploying. (Bundles require storage space of about two and a half times their size.)

Memory Requirements: When a bundle is being deployed to a client machine, the total memory required for that session includes the size of the following:

- the base footprint of the JVM being used for the installation,
- the footprint of the DDCAM,
- the footprint of the application being deployed.

While the memory requirements of the DeployDirector components are small, requirements are once again heavily dependent on the bundles that you plan on deploying (specifically, the bundle and required JRE memory needs).

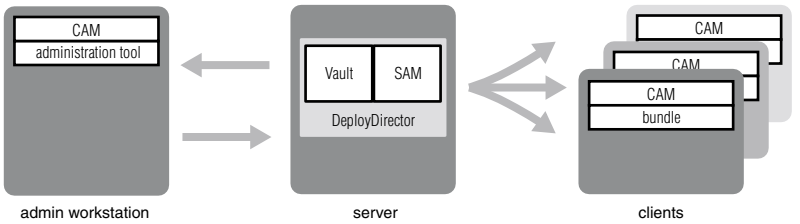
An Overview of DeployDirector Installations

DeployDirector consists of several core components, all of which need to be installed whether you are setting DeployDirector up in a test or production environment. It is the environment in which DeployDirector is being tested or used that determines *where* specific components are installed and configured.

The Server-side Application Manager (SAM) is the servlet that directs traffic to and from an application server. As such, it can either be installed and configured to run with the bundled standalone server, or on top of a commercial server. (The bundled standalone server can act as a server in both a test environment, as well as an actual deployment network.)

The Client-side Application Manager (CAM) resides on client-side machines, and works with the SAM to receive bundles. This component is automatically installed on any client machine during the deployment of a bundle.

The Administration Tool is used to manage server-side activities, particularly bundle maintenance. The tool is meant to be installed on an administrator's workstation, and can either be set up via the DeployDirector installation CD, or can be deployed as a bundle.

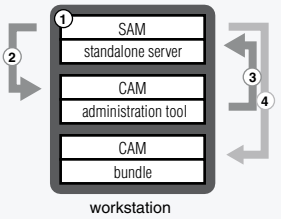


Whether this installation is a test run or for production, installing DeployDirector involves:

- unarchiving DeployDirector onto a server (a test workstation running the standalone server, or a production server running the standalone server or another application server),
- verifying and changing default configuration settings,
- deploying the Administration Tool (which includes automatic deployment of the CAM) to a designated administrator's workstation.

The following diagrams outline typical deployment environments. They in turn determine where DeployDirector components are installed (steps 1 and 2), and where deployment is managed, and occurs (steps 3 and 4).

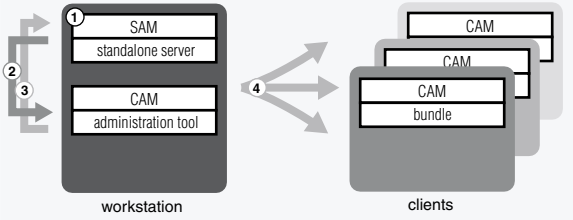
Model 1: Installation on, and deployment to single test workstation.



On a test workstation, DeployDirector is installed and the standalone server is configured as the deployment server (1). The administration tool is deployed to the same workstation (2), and is used to access the SAM (3). Test bundles are deployed to the same machine (4).

In this model, the same machine acts as application server, administrator's workstation, and client machine.

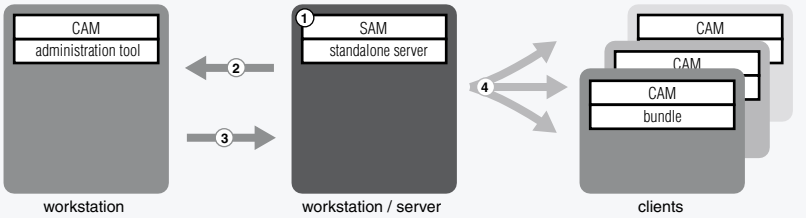
Model 2: Installation on test workstation, deployment to true clients.



On a test workstation, DeployDirector is installed and the standalone server is configured as the deployment server (1). The administration tool is deployed to the same workstation (2), and is used to access the SAM (3). Bundles are deployed to any number of clients (4).

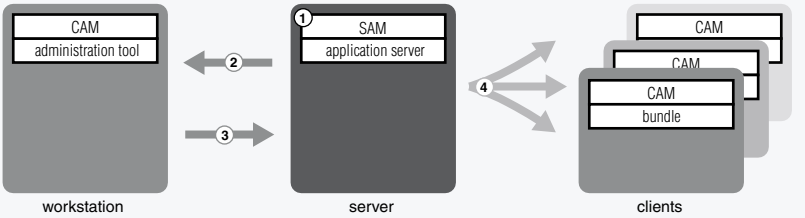
This model allows mass test deployments to a network of clients, while maintaining simplicity by allowing the same machine to act as both server and admin workstation.

Model 3: Installation on test or production server with Tomcat, deployment to true clients.



Used as a realistic test or true production environment, this model consists of a SAM and admin tool installed on (1), and deployed to (2), their own dedicated machines.

Model 4: Installation on, and integratation with commercial application server, deployment to true clients.



This setup is identical to the previous model, with the exception that DeployDirector is running as a servlet on top of a commercial application server instead of the bundled standalone server.

Installing DeployDirector to Your Server

The DeployDirector CD includes automated installers for various supported platforms, as well as archives from which manual installations can be performed (`deploydirector.zip` and `deploydirector.tgz`).

The following procedure covers the general installation of DeployDirector with the standalone server via the automated installer.

Note: Situations that call for a manual DeployDirector installation typically involve its use as a servlet engine with an application server.

1. Insert the DeployDirector installation CD into your CD-ROM drive, and wait for auto-run feature to begin. Alternatively (or if you are using a downloaded evaluation version), locate and run the installer for the appropriate platform.

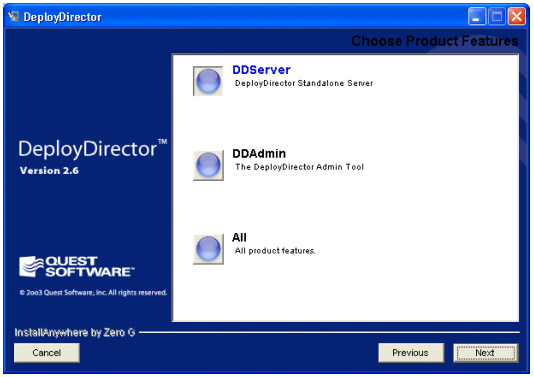
   For Windows: If the auto-run feature does not initiate the installation process automatically, choose Start > Run. Locate the `windows_dd_250.exe` file at the root of the DeployDirector CD, select it, then click OK to begin the installation.

   For Unix: Run the `bin` file whose name matches the platform on which you are installing DeployDirector. These files are found at the root of the DeployDirector CD.

   Note: In order to mount a CD in HP-UX, you will have to enter the following commands:
   `su` (This will put you in the "super user" mode. You will need to supply the root password.)
   `mkdir /cdrom` (You probably want to add this directory at the root of the drive.)
   `mount -F cdfs -o cdcase /dev/dsk/cdrom_device /cdrom` (where `cdrom_device` is listed in the output of the `ioscan -f -n` command.)

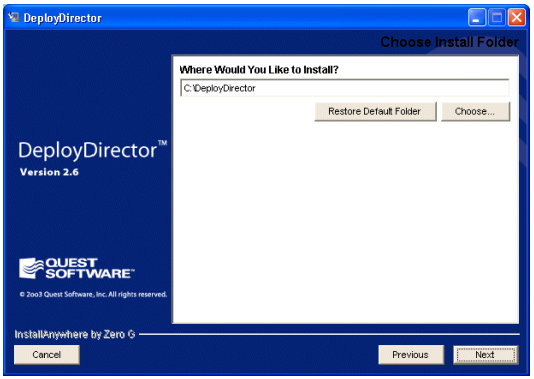
   To unmount the CD, you will need to enter the following command:
   `unmount /cdrom` (where `/cdrom` is the location where you mounted the CD).

When the setup program runs, after accepting the terms of the License Agreement, you are asked to select which product feature you want to install:



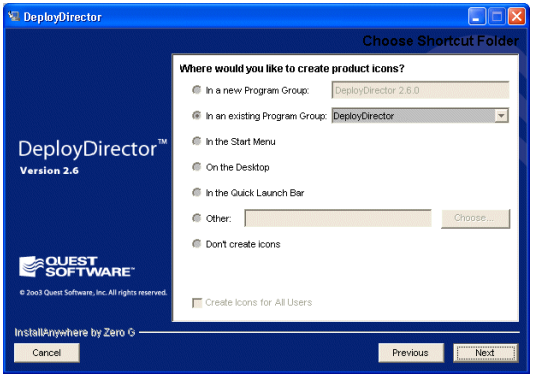
2. Select DDServer, then click Next to begin installing the DeployDirector Standalone Server.
3. Confirm or modify the base install directory, then click Next.

   All of the DeployDirector server-side components will be installed in this location.



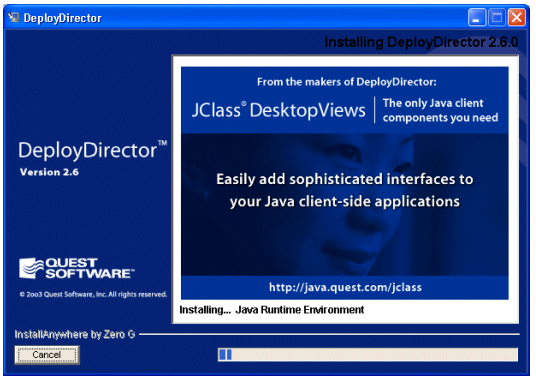
4. Indicate what type of shortcuts you would like to have created.

   Links to all relevant items, including the Administration Tool and documentation, will be placed in the chosen group.



5. Review the Summary, then click Install.

   The server-side DeployDirector components will be installed based on the choices made during this installation process. Once the installation process has completed, the DeployDirector install wizard will hand off duties to the Configuration Wizard.



Configuring and Running the Server-Side Component

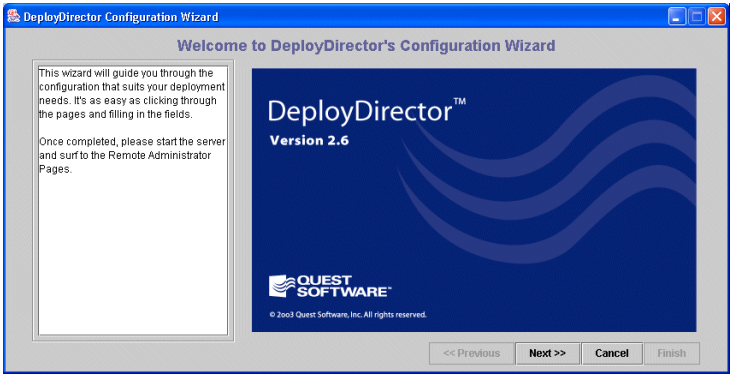
If you plan to use DeployDirector with the included standalone server, you can use the Configuration Wizard to easily enter information about the machine on which the standalone server is running. If you plan on integrating DeployDirector with another application server, you can contact technical support for the latest information on using DeployDirector with supported application servers.

Configuring and Running the Standalone Server

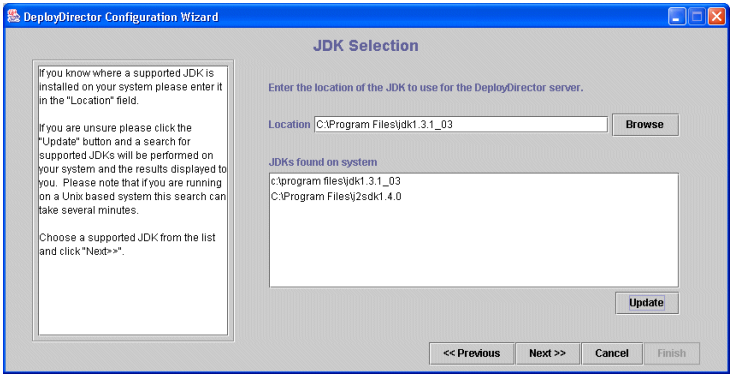
The standalone server is pre-configured, installed with your distribution, and is an excellent way of quickly establishing your deployment network, or a test environment.

Before running the standalone server, you need to configure settings that indicate the host name and port of the server on which it is running. This is done quickly and easily with the Configuration Wizard. When DeployDirector is successfully installed, you will be asked whether you wish to run the Configuration Wizard. The text field in the left pane of the Configuration Wizard provides relevant explanations at each step.

If you choose to run the Configuration Wizard, it will guide you through the configuration process.

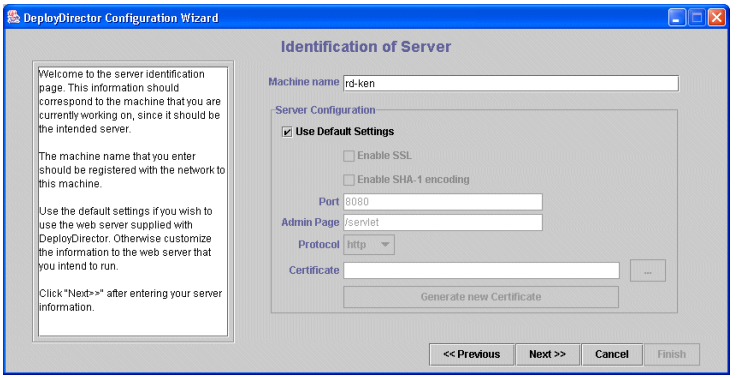


The Configuration Wizard first prompts you to indicate which JDK is to be used with the DeployDirector server. Click Update to have the Wizard list all found JDKs on the machine on which DeployDirector has been installed:

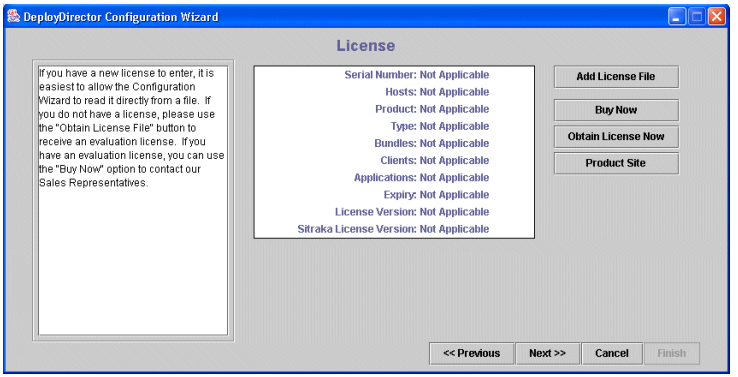


The Wizard next prompts you for the name of the machine as identified on the network, along with the port setting, and preferred protocol. If you wish to enabled SSL, you will require a certificate, which you can either designate, or generate.

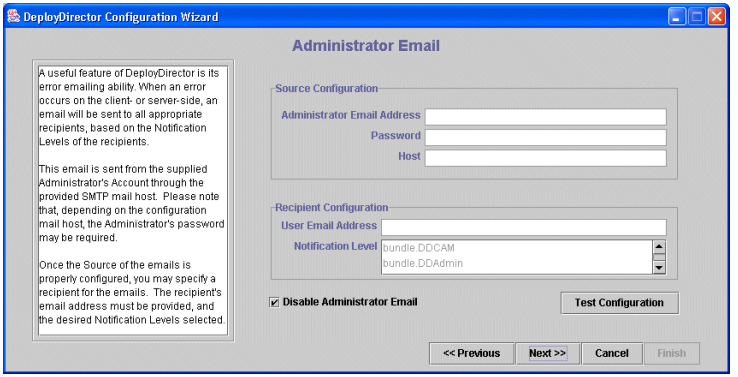
You will also be able to enter the path to the Administrator's Page (`/servlet/` is the default setting), and indicate whether you want advanced server encoding to be used.



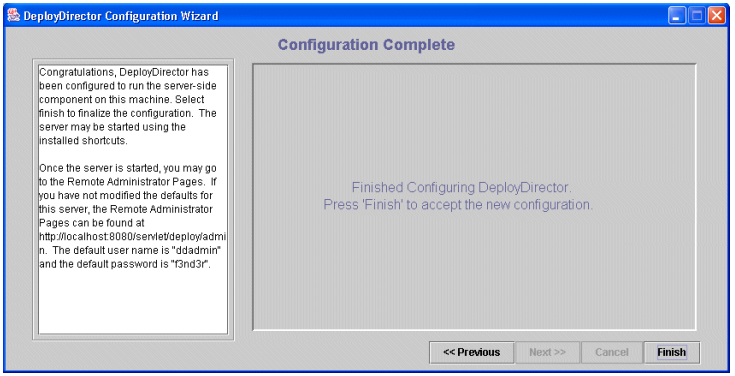
You are now required to enter the license information provided by Quest Software. If this is a temporary license for evaluation purposes, please note that you will subsequently be able to update license information from the Remote Administrator in your browser.



Error emailing is another useful feature provided by DeployDirector. If an error occurs on the server or client side, the network administrator will be notified if this is enabled in the Configuration Wizard. (You still can modify error emailing lists after DeployDirector has been installed, through the Remote Administrator.)



Once all of these details have been configured, you may review them all in the Configuration Summary screen. After confirming these settings, the configured standalone server will be ready to run.



The server can be started by using the startup shortcut that was created, or by executing the startup batch file found in the standalone server directory of the DeployDirector installation.

Running DeployDirector as a Servlet Engine with an Application Server

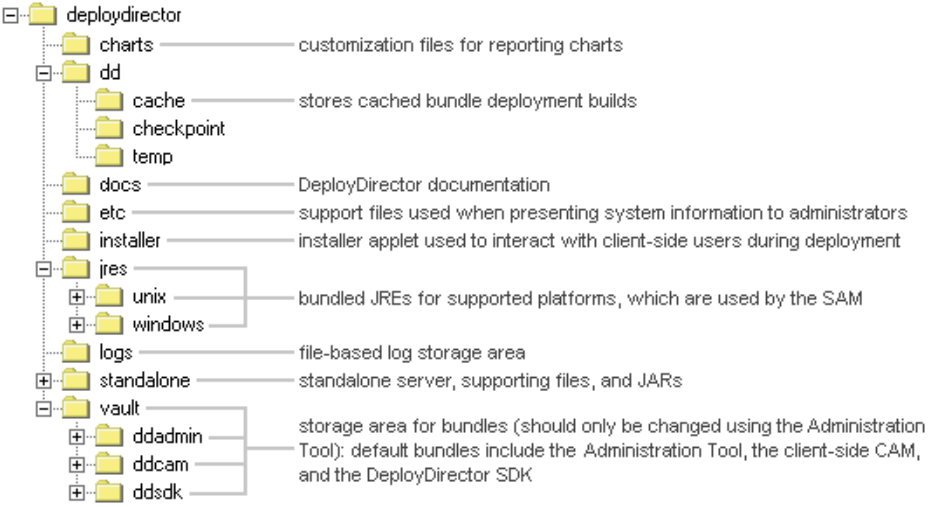
While the standalone server offers a quick and easy way to establish your deployment network, you may want the added configuration benefits of a commercial Web server, with which DeployDirector can function as an add-on servlet engine.

Supported servers are listed in Supported Platforms and General Requirements, found earlier in this chapter. For the most up-to-date information on integrating DeployDirector with any of these Web servers, please contact DeployDirector Technical Support:

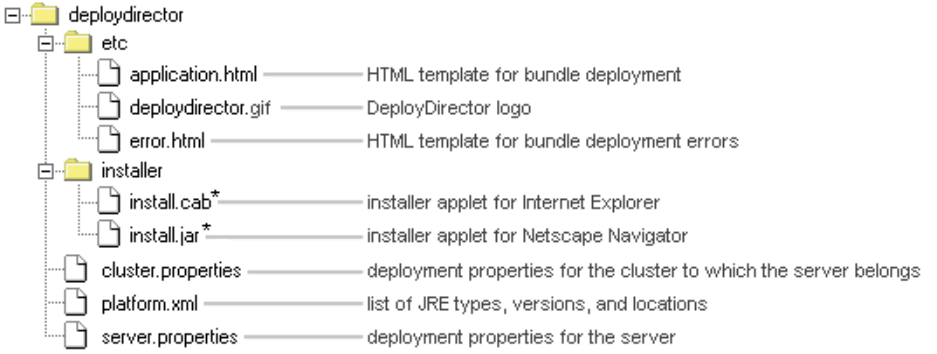
Quest Software Web Site (Java Products, DeployDirector Support):	http://java.quest.com/support/deploydirector/
North American Support Information:	dd_support@sitraka.com 800-663-4723 (toll free in North America) or 416-594-1026 Monday to Friday, 9:00 a.m. to 8:00 p.m. EST Fax: 416-594-1919
European Support Information:	Email: eurosupport@sitraka.com Phone: +31(0)20 510 67 00 Monday to Friday 9:00 a.m. to 5:00 p.m. CET Fax: +31 (0)20 470 03 26

Installed Directories and Location of Key Files

The following outlines the contents of the DeployDirector installation:



Once DeployDirector has been installed, you may want to work with individual files to customize DeployDirector to match the needs of your organization. The following outlines the locations of the key files that you can customize, or on which your deployment process will depend:



* If you are deploying bundles for non-evaluation purposes, you will need to re-sign the installer with your own organization's certificate. By default, a Quest Software certificate is used.

Accessing the Remote Administrator to Enter Your License

The Remote Administrator gives you control over any DeployDirector-enabled server on the network. While its primary use is for server and cluster maintenance, as well as viewing logs, you can also use it to view and change your DeployDirector license.

Whether you are evaluating or have purchased DeployDirector, a member of Quest Software's sales or technical support team will have provided you with a license to use.

Note: At the time you enter this license, it is also a good idea to change the administrator password.

In a previous section, you configured and began running the standalone server. (If you have since shut it down, please restart it.) You can access that server with the Remote Administrator from any Web browser.

Invoking the Remote Administrator:

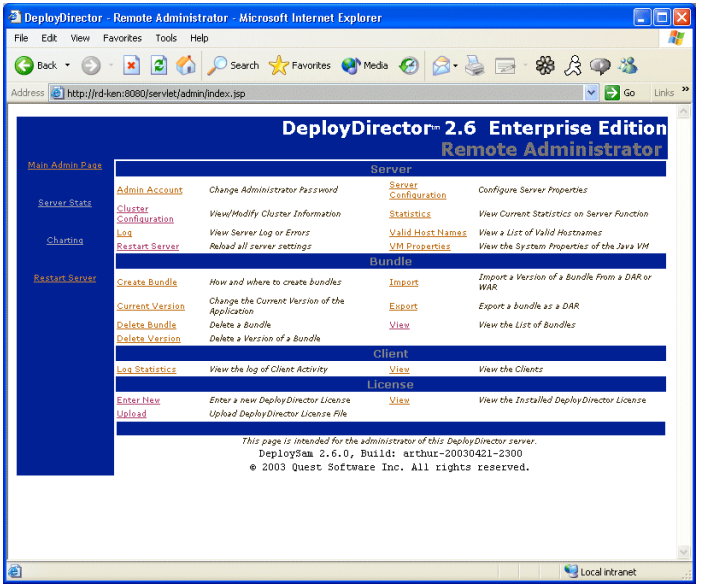
1. Ensure the server on which DeployDirector was installed is running.
2. In your Web browser, load the Remote Administrator by entering this URL:

   `http://[your_host_name]:8080/servlet/admin/index.jsp`

   In this address, `[your_host_name]` indicates the host name property you set in the Configuration Wizard in the previous section. This distribution uses port 8080, but if your server is configured to use another port, use that number instead.

   Please note that the `/servlet/` path is a default setting which refers to the path for the Remote Administrator, set in the Configuration Wizard in the previous step.

3. When prompted, enter your administrator user name and password to load the main Remote Administrator page. (The default user name entered is `ddadmin`, and the default password is `f3nd3r`.)

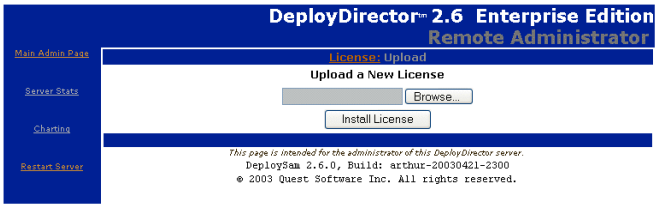


This is an HTML-based front end with which deployment network information can be accessed, and settings can be made. In this case, you want to enter a new license, as well as change the administration password.

Changing the DeployDirector license:

1. From the main Remote Administrator page, click the Upload link.

   The License: Upload page appears, prompting you to enter the path, or browse for the license file provided by Quest Software.



2. Enter the path, or navigate to the license file.
3. Click Install License.

   The Remote Administrator confirms that a valid license has been entered.

4. Click the Restart Server link. Once restarted, click the Main Admin Page link to return to the main Remote Administrator Page.

   Now that you have successfully entered your license, and restarted your server, you can change the administrator password used to access the Remote Administrator.

Changing the administrator password for the Remote Administrator:

1. Ensure you are viewing the main Remote Administrator page.
2. Click the Admin Account link.
3. Enter `f3nd3r` as the old default password.
4. Enter your new password information.
5. Click Submit and wait for confirmation that the new password has been accepted.
6. Click the Restart Server link. Once restarted, click the Main Admin Page link to return to the main Remote Administrator Page.

   Now that the password has been changed to your own, all administrative functions have been taken care of, and you now can begin to deploy DeployDirector bundles to complete the installation of the product.

Deploying the Administration Tool to a Workstation

Once the server on which DeployDirector was installed is ready to serve up data (whether driven by the standalone server, or another application server), the Administration Tool can then be deployed to a system administrator's workstation. While the Administration Tool is an application bundle that only system administrators will use, its underlying deployment process is identical to any other bundle your organization will deploy.

The Administration Tool can be deployed to a system administrator's workstation by using the Remote Administrator. If you have just completed the steps outlined in the previous section, your Remote Administrator is connected to the server on which DeployDirector was installed.

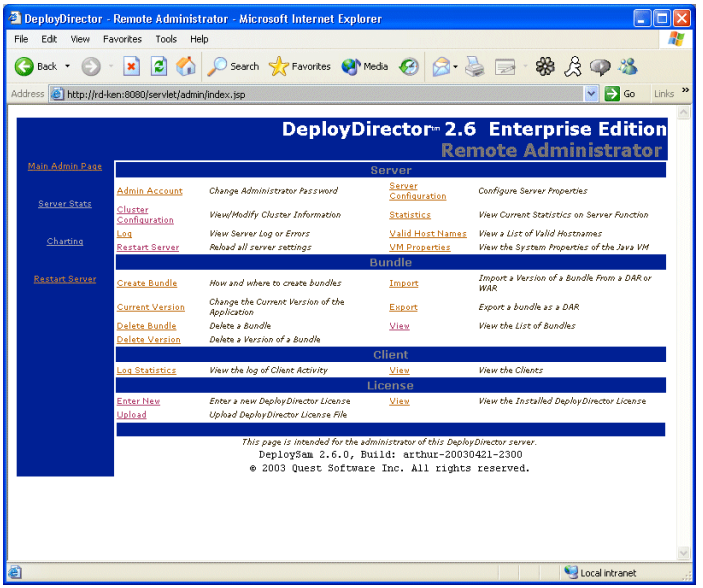
The workstation you use to connect to DeployDirector on the server is the one on which you want to install the Administration Tool. If you are running the Remote Administrator on the desired workstation, please skip to step 4, otherwise, follow the steps as described.

1. Ensure the server on which DeployDirector was installed is running.
2. From the machine on which you wish to install the Administration Tool bundle, enter this URL in your Web browser:

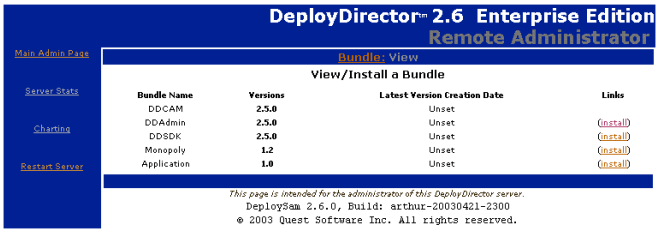
   `http://[your_host_name]:8080/servlet/admin/index.jsp`

   In this address, `[your_host_name]` indicates the DeployDirector host name property you set in the Configuration Wizard previously in this installation. This distribution uses port 8080, but if your server is configured to use another port, use that number instead.

3. When prompted, enter your admin user name and password to load the main Remote Administrator Page.



4. Navigate to the Bundle: View page to display all the bundles available for deployment from the server.



5. Click the install link for the DDAdmin bundle.

6. Instead of navigating to the DDAdmin bundle install page, you also can enter this URL:

   ```
   http://[your host name]:8080/servlet/deploy/ddadmin/install
   ```

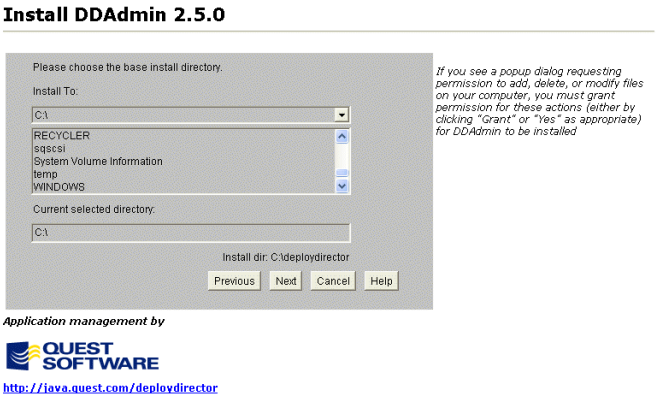
   This URL follows the DeployDirector convention for sending bundle requests to the SAM from a client browser. The `deploy/ddadmin/` portion indicates that a bundle is being requested, and that bundle is the Administration Tool. The `/install` portion of the URL is a command that triggers the client-side installation routine for that bundle (a routine whose files are found in the `install.jar` or `install.cab` file).

   When a download and install request is generically sent (i.e. no specified version number is requested) as with this example, the SAM refers to the `versions.lst` file (created and found in all main bundle directories). From this file, the version listed at the top of the file is deployed.

7. Accept the signed certificates.

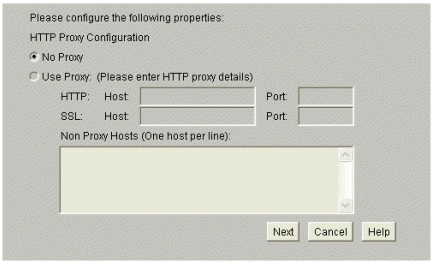
8. Accept the terms of the license agreement.

9. Select or create a destination directory on your workstation hard drive for the Administration Tool.



Once you have indicated where the Administration Tool is to be installed, the SAM checks to see if a CAM already exists at that installation location. If it does not find one (which is likely, considering you are installing the Administration Tool for the first time), the latest version of the CAM is sent. Until the end of the installation, the CAM assumes responsibility for all deployment actions.

10. If you are using a proxy, enter its host name, the port on which it can be found, and list any host names that can be reached without its use. If your proxy uses SSL, enter the same type of information in the designated fields. This information will be used by the Administration Tool whenever it connects to the server

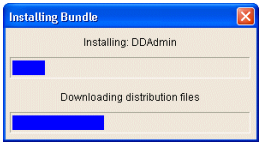


If entering non-proxy host names, ensure that:

- entries are separated with "|"
- host names and IP addresses are valid (e.g. `11.22.33.44|www.quest.com`)
- in order to support the use of wildcards, the host name starts with "." (e.g. entering "`.quest.com`" will match any host name that is `*.quest.com`, such as `www.quest.com`, or `ftp.quest.com`.

11. Indicate whether you want an Administration Tool desktop shortcut to be created.

Now that all the required information has been gathered, the Administration Tool is installed on your workstation.



After viewing the readme, the installation will be complete. You can run the Tool by selecting it from the Start Menu, or double-clicking its desktop icon (if available).

A Note On Supported Browsers

Netscape Navigator and Microsoft Internet Explorer

The browser installed on the client side must be Java-enabled, i.e. the option to install the JVM must be selected during browser installation. Otherwise, either reinstall the browser to include the JVM, or follow the instructions for AOL clients on how to use the Java Plug-in for client-side installation and launching of the bundles.

AOL

For end users: AOL 6.0 clients require the Java Plug-in from Sun Microsystems in order to install and launch applications from the DeployDirector server. The following format should be used instead of the `install` and `launch` requests:

```
http://[server_name]:[port]/servlet/deploy/[bundle_name]/install-plugin
```

```
http://[server_name]:[port]/servlet/deploy/[bundle_name]/launch-plugin
```

If the Java Plug-in is already present on the client side, the end user can install and launch applications from the DeployDirector server. Otherwise, the end user is presented with an option of downloading and installing the Java Plug-in, before proceeding with the installation or launching of the bundle.

For system administrators: Whenever a new version of the Java Plug-in is released by Sun Microsystems, it will be the decision of the network administrator whether to upgrade on the server and client sides. The administrator will need to obtain the following four parameters from Sun Microsystems and modify them in the `cluster.properties` file located in the `<localdrive>/<installpath>/deploydirector` directory:

```
deploy.applet.javaplugin.ie.classid=clsid:8AD9C840-044E-11D1-B3E9-
00805F499D93
```

```
deploy.applet.javaplugin.ie.codebase=http://java.sun.com/products/
plugin/1.3/jinstall-13-win32.cab#Version=1,3,0,0
```

```
deploy.applet.javaplugin.type=application/x-java-applet;version=1.3
```

```
deploy.applet.javaplugin.ns.pluginspage=http://java.sun.com/products/
plugin/1.3/plugin-install.html
```

If the version of the Java Plug-in on the client side does not match the specifications within the `cluster.properties` file, the client will be prompted to install the new version of the Java Plug-in from Sun Microsystems.

Upgrading the DeployDirector Server

The following procedure outlines how to upgrade DeployDirector from a 1.x or 2.0.x installation to 2.5. The following assumptions have been made:

- you are testing the upgrade in a separate-non production environment,
- you are installing DeployDirector 2.5 to a machine that currently has an older version installed on it,
- you will be using the same port as your existing DeployDirector install,
- the Administration Tool for your existing installation is working.

Performing the upgrade manually

1. Close the Administration Tool, and shut down the old server and any of its dependent applications.
2. Install the version of DeployDirector to which you are upgrading on that machine. You will need to back up your existing 1.x or 2.0.x server if you are re-installing to the same location.
3. In the new server installation, modify the `cluster.properties` file to indicate your machine name and anything else required for your setup. This should be very similar to the `cluster.properties` file for your old installation.

   Do not start up the new server yet.
4. Go to the old DeployDirector version's vault directory and copy the `bundles.lst` file and any bundles *other* than DDAdmin, DDCAM and DDSDK (these three bundles are treated differently as outlined in step 6).
5. Paste these old bundles into the new version's vault directory.
6. To copy the DDAdmin, DDCAM and DDSDK bundles, go to the old DeployDirector version's vault directory, and instead of copying the bundle directory (e.g. DDCAM), copy the bundle's version directory (e.g. 1.5.3).
7. Paste the copied version directory into the directory of the appropriate bundle in the 2.5 vault directory.
8. For each of the DDAdmin, DDCAM, and DDSDK bundles, modify the `versions.lst` file in the new vault so that the list now includes the new version number (e.g. 2.5) as an entry above the old version number (eg.1.5.3).

   In the `versions.lst` file, the top version is always the most current version.

9. Start up the new installation of the server. You will have to license the new server using the Remote Administrator, which is covered on page 16.

10. Start up the same Administration Tool that you used before and it should update itself to the new version.

Performing the upgrade using DAR import and export commands

1. Start up the old server (i.e. the DeployDirector version that is being replaced).

2. Export all versions of every bundle you wish to migrate to the new system.

3. Shut down the old server, and any of its dependent applications (including the Administration Tool).

4. Install the version of DeployDirector to which you are upgrading on that machine. You will need to back up your existing 1.x or 2.0.x server if you are re-installing to the same location.

5. In the new server installation, modify the `cluster.properties` file to indicate your machine name and anything else required for your setup. This should be very similar to the `cluster.properties` file for your old installation.

6. Start up the new installation of the server. You will have to license the new server using the Remote Administrator, which is covered on page 16.

7. Once the server is running, use the Remote Administrator or the Administration Tool, use import the DARs you have created.

Post-Installation Notes

- To test this new installation, you can start up one of the client application bundles you had previously installed via the older version of DeployDirector. The CAM for this application should be updated and the application should start up.
- It is strongly recommended that you upgrade the server and let the clients connect so that a CAM update can occur *before* you publish a new version of the application bundle to the server.
- You may want to note any dialogs that appear as the CAM is updating. It may help you to document what the customer experience will be.
- If any of the assumptions above do not apply to your situation, or you feel that your environment requires special upgrade considerations, please contact Quest Software Technical Support. Contact information can be found in page 39.

General Upgrade Practices

If you are installing DeployDirector over a previous installation (whether or not it is an older version), to ensure a smooth transition to a new installation, please ensure you have:

- deleted all previous DeployDirector bundles,
- saved a copy of your `cluster.properties` file, located in the root `deploydirector` directory, which contains your licensing information (this file can be recopied over the new `cluster.properties`, which saves you the trouble of re-licensing your copy of DeployDirector),
- deleted all instances of DeployDirector from your server.

Previous customers can use the transfer commands in the Administration Tool to move the vault, authentication and authorization data, configuration files, and (if compatible) license from an old server to the new one. Information on transfer groups can be found in Transfer Groups in Chapter 3.

Chapter 2
Introduction

Welcome to the DeployDirector Administrator's Guide. Within this guide, you will find conceptual and procedural information about managing and configuring bundles for deployment, as well as maintaining and administrating your deployment network.

Overview of the Administrator's Guide

The concepts in this guide are presented with the assumption that you have already consulted Chapter 1, Installation and Setup, for setup information and product orientation. It offers all the required information to set up a Server-side Application Manager on your deployment server, as well as the Administration Tool on a system administrator's workstation.

This chapter introduces several basic concepts for use with, or about, DeployDirector or the Administration Tool. Once you have read through this chapter, you should have the knowledge required to begin setting up your organization's deployment network, for which more detailed information can be found later in this guide.

Chapter 3, Managing Servers and Clusters, begins with an overview of the Server-side Application Manager (SAM) and server-side functions. An introduction to the server-side property files is given, followed by server and cluster configuration procedures using the Administration Tool.

Chapter 4, Adding Bundles and Defining Bundle Content, discusses making changes to the vault, as well as adding files and directory structures to bundles.

Chapter 5, Configuring Bundle Installation Properties, offers concepts and procedures that pertain to the deployment process on the client-side, and the Client-side Application Manager (CAM). This includes an overview of the installer applet and the launcher applet, as well as the configuration of bundle installation properties.

Chapter 6, Configuring Bundle Runtime Properties, provides detailed information about bundle properties that take effect when an application is started. This includes procedures related to JRE requirements, VM sharing, authentication and authorization. Additionally, a brief overview of deployment security issues and implementations is provided.

Chapter 7, Configuring Bundle Update Policies, gives detailed information and examples about setting update and connection properties for bundles.

Chapter 8, Preparing Bundles and Servers for Deployment, discusses how to upload bundles to vaults, and creating installation CDs for bundles. Additionally, information on the DAR command line tool is given.

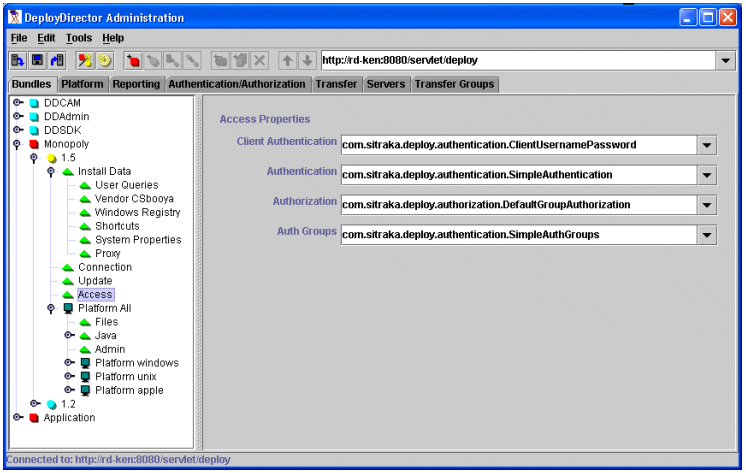
Chapter 9, End User and Administrator Access , gives a full overview of DeployDirector's default authentication and authorization modules/classes and their applications. Building on this knowledge, the use of the Administration Tool to manage administrator roles and end-user access to bundles is covered.

Chapter 10, Viewing and Managing Logs, summarizes DeployDirector's deployment log types, and viewing them in the Administration Tool's reporting tool or the Remote Administrator. Additionally, detailed information about email error reporting is provided, and illustrated with various configuration examples.

Chapter 11, Customizing Functionality with the SDK ,is a users guide for the DeployDirector SDK. This chapter contains programming examples meant to complement the API information that is part of the SDK bundle.

The Administration Tool

The Administration Tool is DeployDirector's primary bundle management tool. It allows you to configure all aspects of deployment bundles. By using the Administration Tool, you can connect to any deployment server on your network from your workstation.



Once a connection with a server has been established, the contents of that server's vault can be seen and configured in the Administration Tool. Tasks that can be performed with the Administration Tool include:

- updating the server to which you are connected with any changes you make,
- adding and removing bundles from a server's vault,
- creating new bundles, and setting their deployment properties,
- configuring the client-side bundle installer applet,
- selecting which Java Runtime Environments exist on the server,
- viewing deployment logs.

The remainder of this chapter describes how these tasks are carried out with the Administration Tool. When visiting some of these tasks, you will be referred to other chapters in this Administrator's Guide where more detailed conceptual and procedural information can be found.

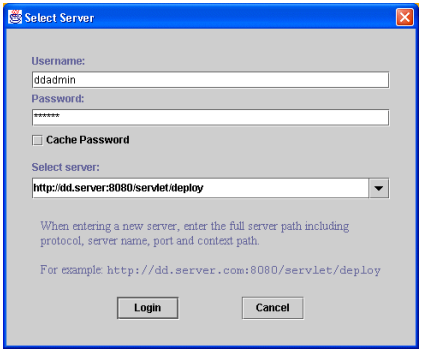
Installing the Administration Tool

The Administration Tool was installed to a system administrator's workstation during DeployDirector installation and setup. You can install the Administration Tool on as many workstations as you require. Please refer to Deploying the Administration Tool to a Workstation in Chapter 1 for more information.

Logging In to the Administration Tool

Whenever the Administration Tool is started, you are required to enter user information, and select a server. When DeployDirector is first installed, the default user name and password (respectively, ddadmin and f3nd3r) can be used, but it is recommended that these are changed using the Remote Administrator, as outlined in Accessing the Remote Administrator to Enter Your License in Chapter 1.

The Administration Tool can connect to any online server on which DeployDirector was installed. Enter the server address, use the drop-down list to locate a previously inputted server.



Updating the Server

When using the Administration Tool to configure bundles for eventual deployment, any bundle changes are initially made locally (i.e. your workstation's hard drive). Only when you update the server with the local changes you have made will the server reflect your actions in the Administration Tool.

This process is designed to act as a safeguard against potential errors to which several factors contribute. It is possible, if not typical, that several deployment servers exist as part of your organization's deployment network. System administrators can access the contents and properties of any server at any time. Additionally, any changes made to a server are replicated across the deployment network to ensure that client-side users have access to the same bundles, no matter which server deploys them. Thus, in order to avoid the ill effects of two system administrators simultaneously modifying vault contents on different servers that are part of the same cluster, modifying the contents of the vault is performed on a local machine. Additionally, it is important that changes being made to a bundle be completed before committing them to a server.

For more information about bundle replication, please refer to Chapter 3, Managing Servers and Clusters.

Committing changes to the server

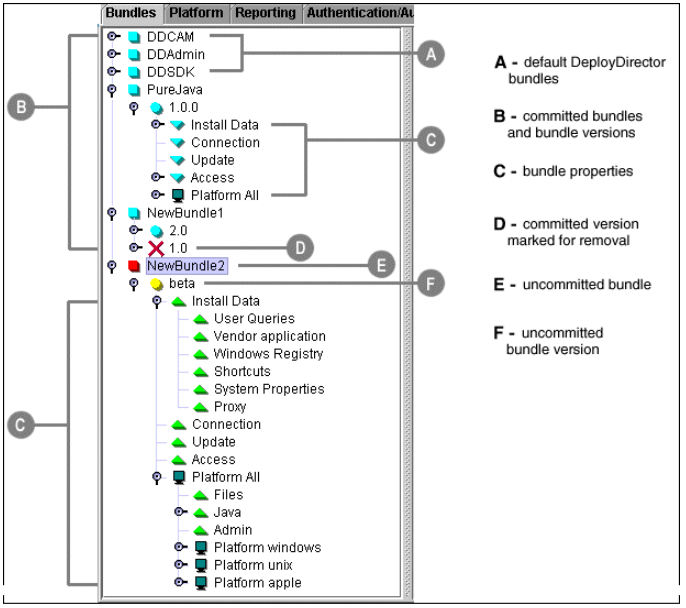
1. Ensure all changes you have made in the Administration Tool are correct and complete.

   Once changes have been committed to the server, they will be replicated across the deployment network to all other servers. Thus, it is critical that changes you make are accurate. This is particularly important with bundle versions you create since they cannot be modified (only removed) after being uploaded to the server.

2. Click File > Update Server.

Working with Bundles

One of the more common tasks performed with the Administration Tool is changing the contents of the vault. The Administration Tool simplifies the task of keeping track of bundle versions and their deployment properties. In the Bundles tab, the contents of the vault on the server to which you are currently connected are displayed:



For more information about working with bundles and their contents, please refer to Chapter 4, Adding Bundles and Defining Bundle Content.

Administration Tool Date and Time Entry Formats

When configuring bundle properties, as well as some server and cluster properties, you will encounter property fields that accept time values. There are many formats that are accepted and correctly interpreted by DeployDirector. The following lists outline these formats.

Abbreviation	Value	Example
MMM	month	OCT
MM	month	10
dd	day	28
yyyy	year	1973
hh	hours	11
mm	minutes	59
ss	seconds	45
z	time zone	EST
a	a.m./p.m. marker	pm

Date and Time Formats	
yyyy.MM.dd hh:mm:ss z	yyyy.MM.dd hh:mm:ss
yyyy.MM.dd hh:mm z	yyyy.MM.dd hh:mm
yyyy.MM.dd h:mm:ss a z	yyyy.MM.dd h:mm:ss a
yyyy.MM.dd h:mm a z	yyyy.MM.dd h:mm a
yyyy/MM/dd hh:mm:ss z	yyyy/MM/dd hh:mm:ss
yyyy/MM/dd hh:mm z	yyyy/MM/dd hh:mm
yyyy/MM/dd h:mm:ss a z	yyyy/MM/dd h:mm:ss a
yyyy/MM/dd h:mm a z	yyyy/MM/dd h:mm a
MM/dd/yyyy hh:mm:ss z	MM/dd/yyyy hh:mm:ss
MM/dd/yyyy hh:mm z	MM/dd/yyyy hh:mm
MM/dd/yyyy h:mm:ss a z	MM/dd/yyyy h:mm:ss a
MM/dd/yyyy h:mm a z	MM/dd/yyyy h:mm a

Date Formats	
MMM dd, yyyy	yyyy/MMM/dd
yyyy.MM.dd	yyyy/MM/dd
MM/dd/yyyy	

Explicitly stated time intervals can also be entered in some property fields. They can be specified as:

value [units]

Acceptable units include seconds, minutes, hours and days. If no units are specified, milliseconds are used.

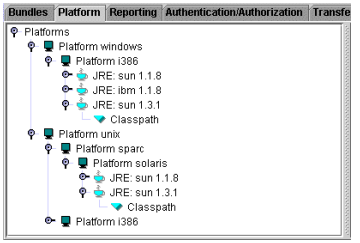
Explicitly stated times are also acceptable:

`now`: the current time
`today`: midnight of the current day
`tomorrow`: the current time, with the next day's date.

Defining Server-Based JREs

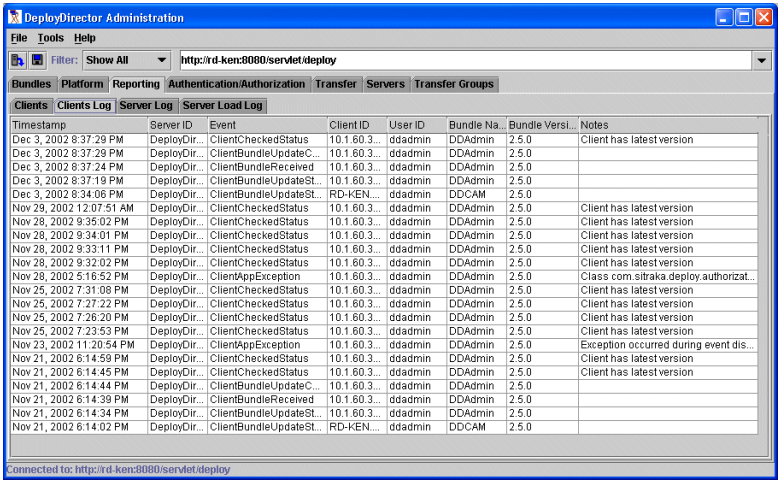
When deploying application bundles, there is no guarantee that the end user's client machine will be equipped with the Java Runtime Environment the application requires. By default, if the client machine does not have the correct JRE it is downloaded from the deployment server.

As such, all JREs used by your bundles must be located on the server side (in the vault). In the Administration Tool's Platform tab, the hierarchical list of all JREs on your servers that are available to be deployed to clients is listed.



It is here that you define the structure of platforms, JRE versions, and locations of those JREs. This list is used when you set the Java property in a bundle, which, if it does not exist on the client machine, will go down this list to find out where on the server it can find the proper JRE.

Viewing Deployment Logs

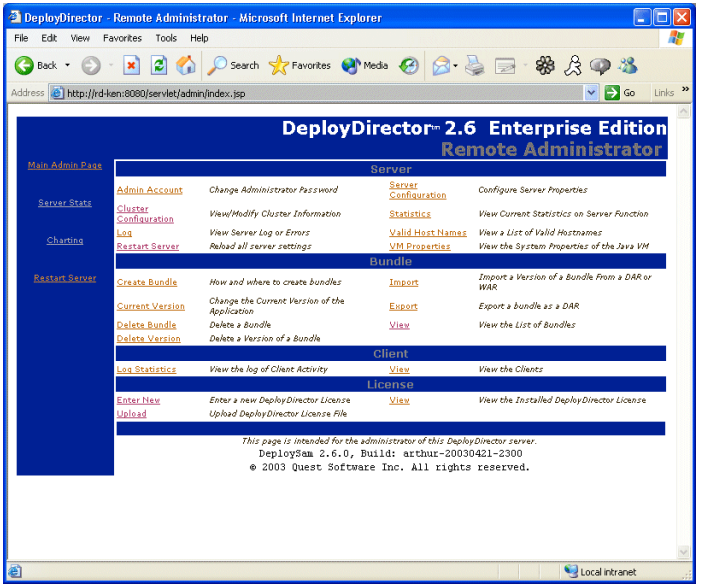


During deployment activity, logs are automatically generated and recorded either as flat files on the deployment servers or in a central database. The Administration Tool contains a reporting tool (accessed by clicking the Reporting tab) in which each of the four logs can be viewed.

For conceptual information about log types, as well as procedural information on configuring your deployment network's logging behavior, please refer to Chapter 10, Viewing and Managing Logs.

The Remote Administrator

While the Administration Tool serves primarily as a bundle configuration tool, the Remote Administrator provides you with higher level control over your deployment network, particularly server and cluster configuration and management. When a stand-alone deployment server is running, entering `http://[your host name]:8080/servlet/admin/index.jsp` in a browser calls the Remote Administrator:



This page provides information on the following areas:

- information about your cluster, including host and public server names
- information about the server to which you are connected
- the bundles available on the server
- bundle management on the server to which you are connected
- access to server and client logs
- your organization's DeployDirector license
- deployment statistics and status for the server
- the server's cluster profile
- information on the build that created the server.

Additionally, actions that can be performed from this page include:

- changing of administrator passwords
- restarting the server
- entering new license information.

Introduction to the CAM

The CAM is the DeployDirector component whose function is managing bundles on the client side (its counterpart, the SAM, is discussed in Chapter 3, Managing Servers and Clusters). If the bundle is a Java application, the CAM oversees its installation, execution and with newer bundle versions, its updating. If the bundle is not an application, but a collection of support files, the CAM still oversees its installation and updating. With this in mind, a CAM must exist on every client machine, and is always sent along with a deployed bundle if it does not yet exist on the client side.

CAM Roles

Establishing contact with the server When a client-side user requests a bundle, or the bundle they are using requires a connection to the SAM, the CAM attempts to make contact with a SAM. The CAM searches for available servers across the network through the configured port (the default HTTP port is 80, and the standard DeployDirector port is 8080). Upon finding one, a connection is established with it.

Establish the validity of the user Once contact has been made with a server, the SAM requires that the user's validity (at least their validity as a bundle user) is established through the CAM. This prevents unauthorized users from accessing bundles, and allows authorized users to do so from different client machines. The two steps that constitute this process are authentication and authorization. (This is discussed in greater detail in Chapter 6, Configuring Bundle Runtime Properties.)

Administrate bundle installations When a user's bundle request has been accepted, the CAM handles all file transfers between server and client. Once the install applet has downloaded the CAM and the JRE with which it runs, it begins to visually walk users through the installation. At this point, it is the CAM that alters and writes to the client file system. As such, the CAM oversees bundle downloads, installations, as well as bundle removal. (You can learn more about the installation process and components in Chapter 5, Configuring Bundle Installation Properties.)

Detect and act on a bundle's properties during installation Bundle properties, which are permanently set before the bundle is uploaded to a server, can vary greatly between bundles and bundle versions. Generally, bundle properties determine what the bundle contains, and how they and the entire bundle are handled during and after deployment. The CAM ensures that all of these are followed through at the client side.

Specifically, the CAM is responsible for detecting and installing:

- platform-specific files within a bundle (e.g. Windows `.dll` files, Start Menu icons)
- platform-specific settings required for the bundle to run successfully (e.g. Windows registry entries)
- execution files for specific platforms (namely, `.exe` and `.sh` files), which are created by the CAM during deployment, and whose creation is originally indicated by the bundle's Entry Points property.

Manage bundle updating on the client side When multiple versions of a bundle exist, the CAM is in charge of determining when users must upgrade to the next version. It determines this by reading the properties of both the bundle currently in use and those of a newer version detected on the server. Update policies within an organization and for particular bundles can vary, and the CAM ensures that actions determined by properties you set are faithfully carried out. (More information on bundle updates can be found in Chapter 7, Configuring Bundle Update Policies.)

Technical Support

Quest Software Inc. provides two support options for DeployDirector customers: Pre-Sales Technical Support, and Gold Support with Subscription. for more information about these please visit:

http://java.quest.com/support/deploydirector/.

Contacting DeployDirector Support

Any request for support *must* include your DeployDirector product serial number. Supplying the following information will help us serve you better:

DeployDirector:

- serial number
- DeployDirector version number (displayed at the bottom of the Remote Administrator pages)

CAM (Client-side Application Manager) information:

- the `ddcam.config` file, found in the `<vendorname>/lib/` directory
- the `bundle.properties` file for the bundle in question, found in the `<vendorname>/<bundlename>` directory
- the `bundle.properties` file for the CAM, found in the `<vendorname>/lib/` directory
- Web browser vendor and version being used
- Web browser JRE and version being used
- an archive of the client install directory in Zip format (not required, but recommended)

SAM (Server-side Application Manager) information:

- the `version.xml` file, found in the `<installdirectory>/deploydirector/vault/<bundlename>/<bundleversion>` directory
- the `platform.xml` file, found in the `<installdirectory>/deploydirector` directory
- `cluster.properties`, found in the `<installdirectory>/deploydirector` directory
- `server.properties`, found in the `<installdirectory>/deploydirector` directory
- cached DeployDirector files, found in `<installdirectory>/deploydirector/dd/cache`
- cleaned log files, found in `<installdirectory>/deploydirector/logs`

Third party information

- all log files from any third party components, including the Web server, application server, proxy, firewall, and browsers (optional, but recommended)
- a complete description of your Deployment environment, including 3rd party components (e.g. firewall, proxy, or load balancer), as well as vendor and version details for your client and server-side OS and JRE

To ensure prompt assistance, please fill out a DeployDirector Technical Support Form at http://java.quest.com/support/supportwatch/.

Contact information

Quest Software Web Site (Java Products)	http://java.quest.com
DeployDirector Site	http://java.quest.com/deploydirector/
North American Support Information:	dd_support@sitraka.com 800-663-4723 (toll free in North America) or 416-594-1026 Monday to Friday, 9:00 a.m. to 8:00 p.m. EST Fax: 416-594-1919
European Support Information:	Email: eurosupport@sitraka.com Phone: +31(0)20 510 67 00 Monday to Friday 9:00 a.m. to 5:00 p.m. CET Fax: +31 (0)20 470 03 26

A Note About Our Transition

Sitraka Inc. is currently in the process of merging some of its technical infrastructure with Quest Software, Inc. During this transition stage, all of the aforementioned contact information should work. However, if you encounter any problems, please visit http://java.quest.com for updated information.

Chapter 3
Managing Servers and Clusters

While later chapters in this guide cover aspects of the deployment cycle outlined in the previous introductory chapter, proper management of your server-side setup is discussed here. Understanding and maintaining your server side components ensure proper deployment and replication. The core server-side component is the SAM (Server-side Application Manager), whose function is managing the deployment and storage of bundles. The SAM's actions are mainly dependent on server-side and client-side actions that are initiated by system administrators and end users. These actions include: client-side deployment requests, administrator-initiated vault changes, and the logging of deployment network activity.

SAM Roles and Responsibilities

Stores bundles and JREs The vault is the server-side entity that hierarchically stores bundle versions and JREs for deployment to client-side end users. It is the SAM that maintains this area and modifies it in response to system administrator actions using the Remote Administrator and Administration Tool.

Replicates vault changes and logs to other SAMs/servers If your organization uses a cluster of servers for deployment, the SAM ensures that vault changes made on one server are reflected in all other servers. Additionally, deployment logs are also transferred across the cluster in real time.

Validates and maintains secure data Within an organization's network, bundle deployment, as well as vault and log replication, can involve a great deal of communication between servers and from server to client. The SAM ensures that transferred data is both valid and secure. This functionality can alternatively be delegated to your application server environment.

Moderates end user access to vault-based bundles It is likely that all of your organization's end users are not meant to have access to the contents of your vaults. Additionally, it is possible that different groups of users may have different access privileges to different bundles and bundle versions. As such, the SAM oversees the authentication and authorization process of these end users, during which it works closely with the CAM on the client side.

Generates differences between bundle versions When a new version of a bundle is created for deployment, it is possible that only a few changes have been made to a large application. To save time and bandwidth, the SAM, when processing a request for the deployment of a new bundle version, does not deploy the bundle in its entirety; instead, it generates a temporary build file that is based on the differences between the user's current version and the new version.

Delivers bundles to clients The SAM transfers bundles in response to CAM requests. Bundles are received and managed by the CAM on the client side.

Logs all deployment and server activity Managing a large deployment network requires constant monitoring for server-side and client-side actions and problems. This can be an unnecessarily tedious task for system administrators. The SAM automatically generates logs that thoroughly, yet succinctly, report activity across the deployment network in a variety of formats. These logs can be easily viewed in the Administration Tool.

Server-Side Processes

The two main server-side processes can be defined by where user action occurs. On the client side, end-user bundle requests that are sent to the SAM initiate the deployment process. Meanwhile, on the server side, system administrators who make and update bundles initiate the server-side replication process.

The Deployment Process from the SAM's Perspective

Deployment always begins on the client side, when end users connect to a deployment server. Once a connection has been made, whether the user explicitly requests a bundle or their current bundle is configured to require them to download a newer available version, this process involves these steps:

1. Authenticated user information, along with a bundle version request, is sent to and received by the SAM from the CAM.
2. The SAM ensures the authenticated user is authorized to download the bundle version they are requesting.
3. If the user is authorized to access the bundle or bundle version, the SAM initiates the deployment of that bundle. The installer applet installs the CAM, then the CAM is instructed to carry out the installation process. If

the application is detected on the client side, then the launcher applet is executed instead.

4. For the major events that occur on both the client and server sides, log entries are generated by the server. If a local logging configuration is being used, the log entry is kept until the next scheduled aggregation period. If a cluster logging configuration is being used, a JDBC connection is opened, and the log entry is added to the central database.

   In addition to being added to logs, if any client or server-side errors occur during the deployment process, error information, in the form of email reports, are immediately sent to the appropriate people found on DeployDirector's recipient list. (For more information about logging and error email reports, please refer to Chapter 10, Viewing and Managing Logs.)

The Server-Side Management Process

Server-side management consists of any bundle changes made to vaults, as well as cluster or server property changes. These changes are made by system administrators using the Remote Administrator. When an administrator connects to a server with this tool, the following steps occur:

1. The system administrator makes changes to either the vault contents (i.e. bundles), or the cluster or server properties. This could mean adding or removing a bundle, or changing a deployment property at the cluster or server level.

2. The server to which the Remote Administrator is connected is updated with the changes.

3. Any changes to the vault or cluster level properties are replicated throughout the cluster. This ensures consistency in the cluster's deployment behavior and bundle content.

   During any inter-server communication, the security of transmission is dependent on the SSL encryption classes being used with DeployDirector. (For more information about security, please refer to relevant sections in Chapter 6, Configuring Bundle Runtime Properties.)

4. For the major events that occur during this process, log entries are generated by the server. If a local logging configuration is being used, the log entry is kept until the next scheduled aggregation period. If a cluster logging configuration is being used, a JDBC connection is opened and the log entry is added to the central database.

   In addition to being added to logs, if any client or server-side errors occur during the deployment process, error information, in the form of

email reports, are immediately sent to the appropriate people found in DeployDirector's recipient list. (For more information about logging and error email reports, please refer to Chapter 10, Viewing and Managing Logs.)

Bundle and Log Replication

As briefly mentioned in the previous section, any changes to the vault or cluster-level properties are replicated throughout the cluster. This ensures consistency in the cluster's deployment behavior and bundle content.

When an administrator connects to a server using the Remote Administrator, they can perform these types of changes. However, since at any point in time client-side end users can connect to any of the servers in the cluster, it is very important that all servers carry the same bundle versions. Otherwise, chaos may ensue as client-side users try to access bundles that exist on some servers and not on others, or are configured differently on different servers.

As an example of the latter problem, such a predicament can arise if two administrators simultaneously create a new version of the same bundle on different servers. Avoiding an odd situation like this requires good communication among all system administrators. However, DeployDirector's automatic replication helps avoid similar problems.

The Rules of Engagement

While bundle and log replication is automatically performed by SAMs, there are a few things a system administrator can do to ensure consistency across the cluster.

1. There is nothing wrong with fear of commitment. Continually committing (then changing), unfinished bundles to servers increases the chance of end users unknowingly downloading an incorrectly configured bundle. Until the bundle is ready to be used by client-side users, make sure you save bundles-in-progress locally on your workstation (by clicking File > Save).
2. When you are ready to commit, let it be known to everyone. Whenever you have finished making changes, make sure they have been committed to the server to which you are connected (by clicking File > Update Server). Once performed, vault and cluster property changes will automatically be replicated to all other servers.

3. Name your creations wisely. Considering the potential number of client-side users, it is best to give each bundle exclusive and scalable names. This makes long term bundle management easier, and helps avoid the possibility of client-side users downloading different bundles with the same name or version number. As a suggestion, use bundle names that make sense to both system administrators (for archiving and identification) as well as client-side users (for usability).

   Another bundle naming issue that you should be aware involves the use of spaces. While acceptable, spaces in bundle names are handled differently by Netscape browsers, where they should be represented by the `%20` string. When instructing client-side users to manually enter URLs to access such a bundle, or when entering the hypertext reference on an HTML-based front end, be sure to substitute all spaces with this string. For example:

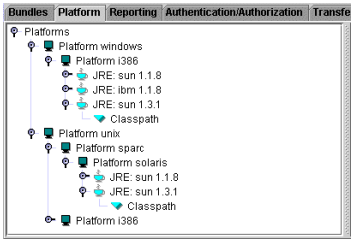
   ```
   http://[your_host_name]:8080/servlet/deploy/bundle name
   ```

   should be entered as:

   ```
   http://[your_host_name]:8080/servlet/deploy/bundle%20name
   ```

JRE Management

When bundles are deployed, the specific JRE required to run the application it contains can also be deployed. This occurs when the bundle's Search for Installed JREs option is enabled and the appropriate JRE is not found on the client side, or when the JRE search option is disabled. (Please see Checking for JREs on the Client Side for more information about using this bundle property.)



The hierarchical list simplifies the cataloging of JREs, as they are categorized by platform, creator, and version.

Adding a new JRE to the vault

1. Ensure the JRE and all of its support files (i.e. the JRE's install directory) have been archived in Zip format.

   It is important to avoid using the distributions provided by the JRE creator (e.g. Sun, IBM), as DeployDirector cannot process them properly.

2. In the Administration Tool, select the Platform tab.

3. Click File > Refresh to load the list of platforms and JREs currently located on the server to which the Administrator Tool is connected.

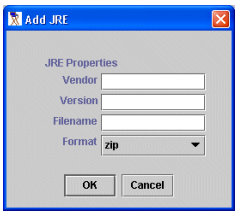
4. If necessary, add a new platform under which the new JRE belongs by selecting its parent in the tree and clicking Edit > Add Platform.

   For example, Platform i386 is added under Platform Windows, and Platform Windows is added under Platforms.

5. In the tree, select the platform to which the JRE belongs and click Edit > Add JRE.

6. In the file chooser that appears, find and select the JRE Zip archive that you created.

7. Selecting the JRE Zip archive displays a JRE properties dialog.



8. In the Vendor field, enter the name of the JRE's creator (e.g. sun, ibm, hp).

9. In the Version field, enter a two-point JRE version number. The two-point version number (e.g. 1.1.8) can be appended, if necessary (e.g. 1.3.0_02).

10. In the Filename field, enter the name of the executable JRE and its full path relative to the root of the JRE Zip archive (e.g. if the selection is `javaw.exe` in the bin directory, enter `bin/javaw`).

11. Select OK.

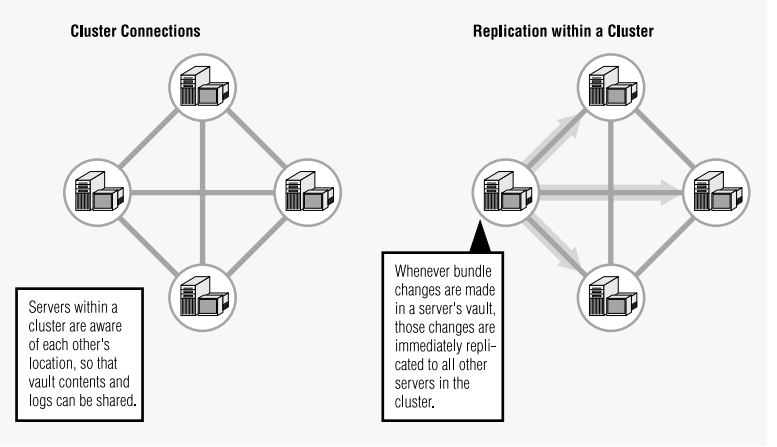
12. Select File > Update Server.

    The JRE is now set up on the server, is reflected in the list in the Administration Tool, and can be referenced during bundle JRE configuration.

Servers and Server Clusters

In an organization with a large base of client machines, using multiple deployment servers alleviates the strain created during heavy client-side download periods (e.g. when all users are downloading a new update within a small time frame). This is accomplished by defining a cluster of servers which are used with a load balancer to evenly distribute requests to SAMs. This type of configuration also ensures that deployment requests can reach a server, even if other servers in the cluster are down.

Outside of actual deployment, clusters also facilitate management of the deployment network. It is important that the list of available bundles is identical to a client-side end user, no matter with which server they have established a connection. As such, SAMs ensure that bundle replication occurs across a cluster whenever any vault changes are made to any one server. (For more control over the sharing of bundles, it is recommended that you set up and use Transfer Groups instead.)



Replication also affects the consistent distribution of activity logs and email error reports. For replication, logging and error reports, it is important that each server in a cluster is aware of the presence and identification of all other servers. This is established through each server's `cluster.properties` file.

Server-to-Server Messages within a Cluster

Internally, servers that are part of a cluster communicate to each other. Certain cues initiate this process. For example, when a system administrator updates a particular server with a new bundle version, the server to which they are connected informs the other servers that a new bundle exists. The other servers then indicate that they are ready to receive the new bundle. In general, servers communicate with each other to inform other servers that:

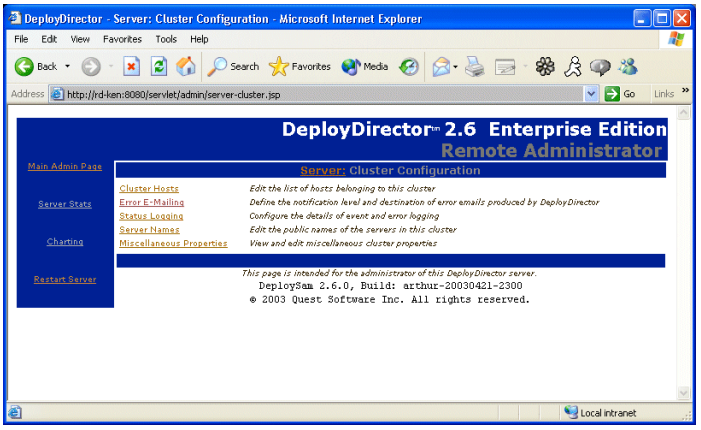
- a new JRE exists
- a new bundle exists
- a new bundle version exists
- a server configuration has been modified
- new log entries have been created
- a new client-side bundle installer has been created.

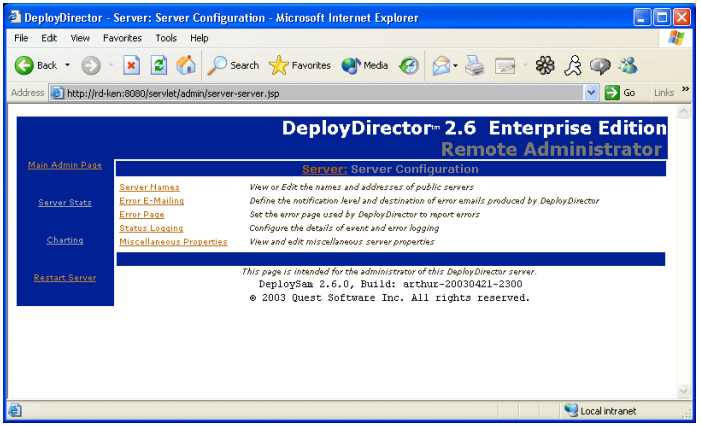
Since these processes are built into DeployDirector, they constitute the tight inter-server communication network that ensures your deployment cluster possesses collective traits that make it easier to manage.

Cluster and Server Properties

Each server possesses a pair of profiles that outline how the server and server's cluster handle deployment, logging and replication. The `cluster.properties` and `server.properties` files define, respectively, how the cluster behaves, and how the server behaves within the cluster. Each server's `cluster.properties` file is identical, while its `server.properties` file can be unique (although this is not always the case).

A server's properties, and the properties of the cluster to which it belongs, are both viewed and edited in the Remote Administrator:





Configuring properties from the Server: Cluster Configuration page in the Remote Administrator affects (at the cluster level) error email processing, logging, and cluster access from the client side.

Although the `cluster.properties` file can be manually edited, it is necessary for you to make cluster property changes with the Remote Administrator. This ensures that changes are replicated across the cluster (which will not happen if you manually change them).

Configuring properties from the Server: Server Configuration page in the Remote Administrator affects, at the server level, error email processing, logging, server access from the client-side and security.

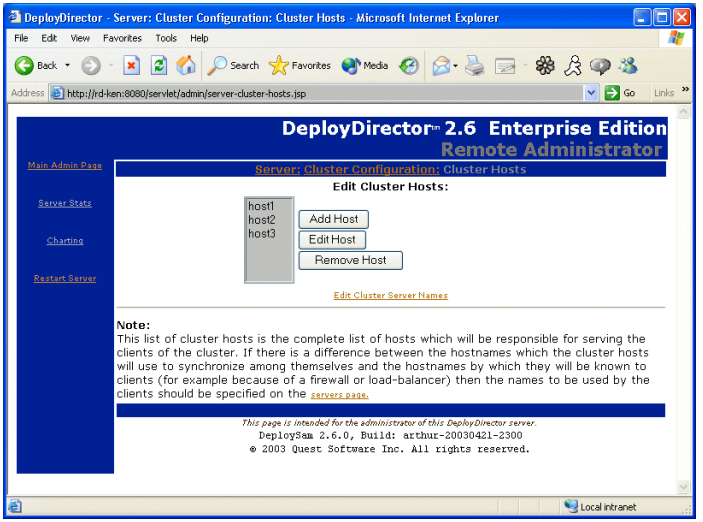
Setting Basic Cluster Properties

Whether or not you set server level properties, you must always define your cluster. Basic cluster configuration tasks include: setting up the cluster, adding servers to the cluster, removing a server from a cluster, and changing a cluster server's properties.

Viewing your server cluster

1. In the Remote Administrator Tool, navigate to the Server: Cluster Configuration: Cluster Hosts page.

   All the servers you have defined in your cluster are listed here:



   It is important to ensure that all deployment servers are listed as hosts here. Otherwise, missing servers will not be part of the replication pool.

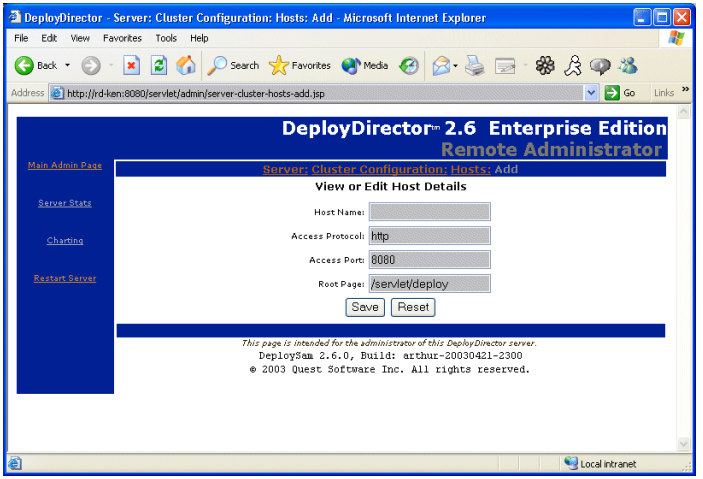
2. Select any server from the list and click Edit Host to view its profile.

The servers listed as part of this cluster deploy bundles and are part of the replication pool. This list also determines which servers generate and receive logs, and send email error reports. From this page, you can also edit the cluster properties to determine which of these servers are visible from client machines. How all of these functions are carried out depends on whether they have also been defined at the server level. These issues are covered later in this chapter.

Adding a server to a cluster

1. In the Remote Administrator Tool, navigate to the Server: Cluster Configuration: Cluster Hosts page.
2. Click Add Host.

   The Server: Cluster Configuration: Hosts: Add page appears, requiring input for several host properties.



3. In the Host Name text field, enter the name by which the server will be known within the cluster. (Please note that this name is not necessarily seen by client-side users.)
4. In the Access Protocol text field, enter the protocol used to access the new server (the default value is `http`).
5. In the Access Port text field, enter the port through which the server is accessed (the default is `8080`).
6. In the Root Page text field, enter the path through which the server's bundles can be accessed from the client side (the default is `/servlet/deploy`).

   The values entered as the machine and page properties constitute the base URL at which the server's bundles are accessed.

   For example, if you entered `installserver` in the Host Name field, and `http`, `8080`, and `/servlet/deploy` in the remaining fields, then the server's access URL is `http://installserver:8080/servlet/deploy`.
7. Review your settings, then click Save.

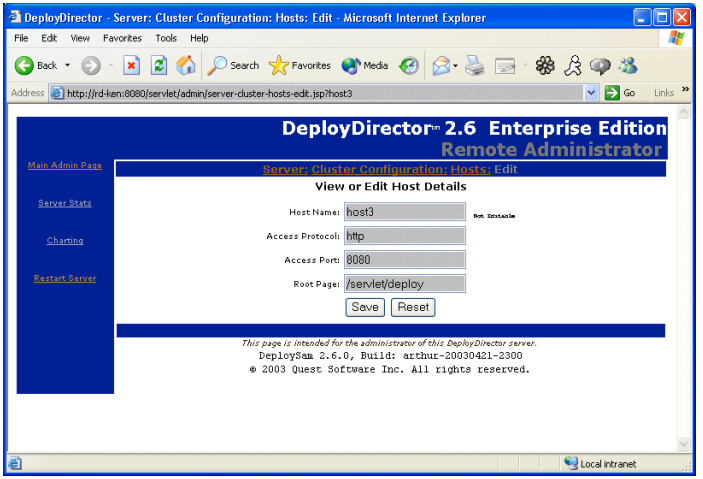
Removing a server from a cluster

1. In the Remote Administrator Tool, navigate to the Server: Cluster Configuration: Cluster Hosts page.
2. From the server list, select the server you want to remove.
3. Click Remove Host, and confirm the action.

   This change is replicated from the server to which you are connected, to all other servers in the cluster. The removed server is no longer part of the replication pool.

Changing a server's host properties

1. In the Remote Administrator, navigate to the Server: Cluster Configuration: Cluster Hosts page.
2. From the server list, select the server whose properties you want to change.
3. Click Edit Host to reveal the server's properties.



4. Change the server's host properties as required.

   The listed property text fields constitute the server's access URL for the user.

   In the above example, since the Host Name is `host3`, the server's access URL is `http://host3:8080/servlet/deploy`.

   The `/servlet/deploy` path is the directory on the machine in which vault-based bundles are accessed.

   Port and protocol determine how you can access the server on the network.

5. Review your settings, then click Save.

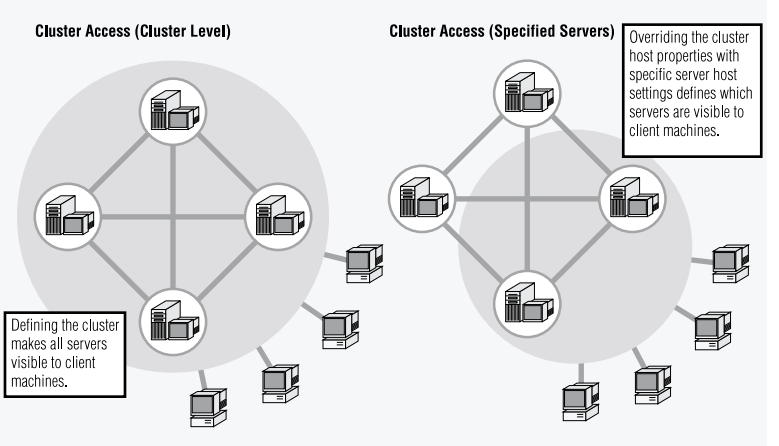
The Combined Effect of Server and Cluster Properties

The configuration of some cluster and server properties are exclusive to that network level. For example, setting the administrator's user name and password is done at the cluster level, and affects only the cluster. Conversely, setting which security class is used is done at the server level and effects only individual servers. However, most properties can be set at both the cluster and server level and have different effects when both sets of properties are combined.

It is important to be aware of the effect of combining cluster and server properties when configuring your deployment network since logging, error email processing, and the definition of visible servers to the client side are all directly affected.

Bundle properties that affect the *logging* of deployment activity, when set at the server level, override those set at the cluster level for that particular server. On the other hand, properties that affect the *sending of email error reports* (conceptually, a subset of logging), when set at the server level, are aggregated with those at the cluster level. (You can find more information about the effects of combining cluster and server properties for logging and error email reporting in Chapter 10, Viewing and Managing Logs.)

Combining cluster and server properties also affects which servers in the cluster are visible to client machines:



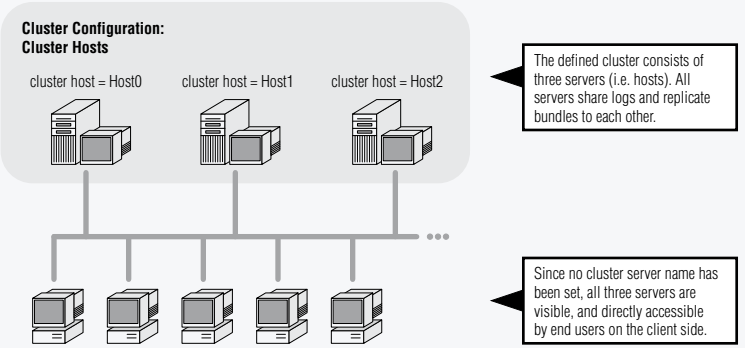
The Client-Side Visibility of Servers in a Cluster

On the client side, end users connect to deployment servers to request, download, and install bundles. Whether this is done directly by the user (by entering a URL into their browser) or through an HTML front end created by your organization (which often displays the name of the server being accessed), by default, all servers in a cluster are visible to the end user.

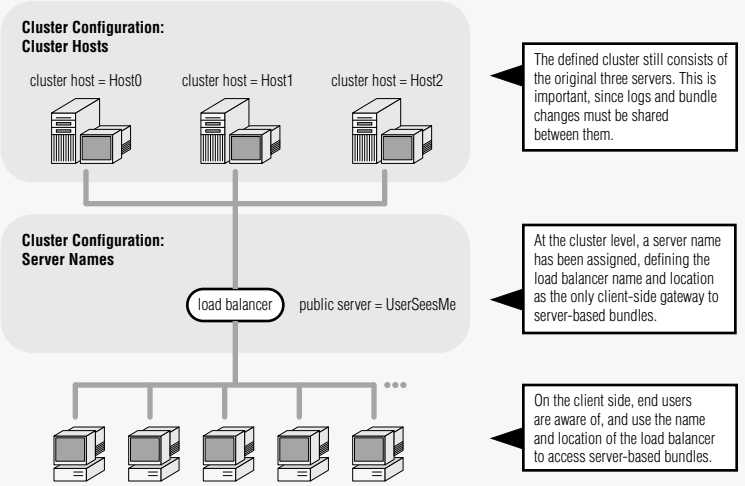
However, there may be cases when you do not want end users knowing about all existing deployment servers. You can easily define, at the cluster level, which servers are meant to be 'seen' by end users. This is done by defining public servers on the Server: Cluster Configuration: Server Names page in the Remote Administrator.

The following example demonstrates how defining public server names in the Remote Administrator can simplify deployment by designating a load balancer as the only visible access gateway to the deployment servers.

The Cluster as End User Access Gateways

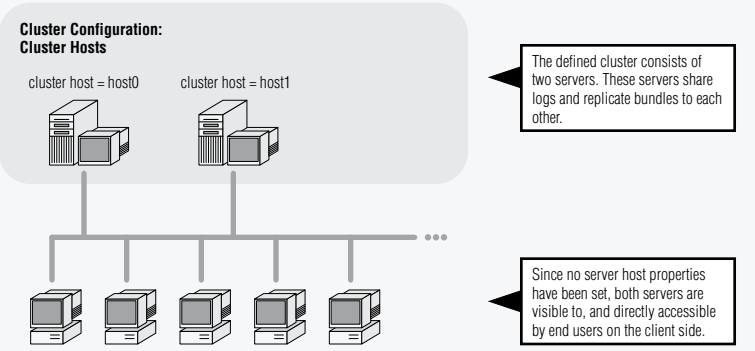


Specifying End User Access Gateways

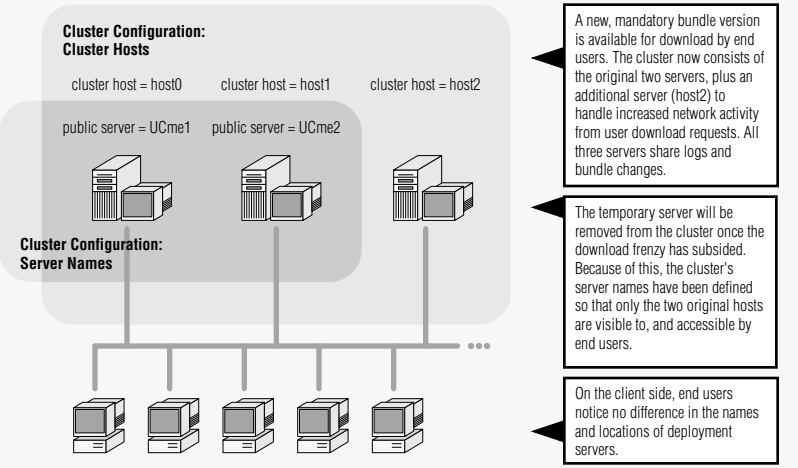


Defining public servers at the cluster level to accommodate a load balancer is the most common application of these settings. However, other situations may warrant similar steps. The following example also demonstrates how defining public servers overrides the cluster-level host settings. In this example, an additional server is temporarily added to the deployment network, and while logging and bundle replication are meant to be shared between all servers, system administrators do not want end users to notice the presence of the additional server.

Business as Usual: the Cluster as End User Access Gateways



Temporarily Using an Extra Server for Increased Network Activity: Specifying End User Access Gateways



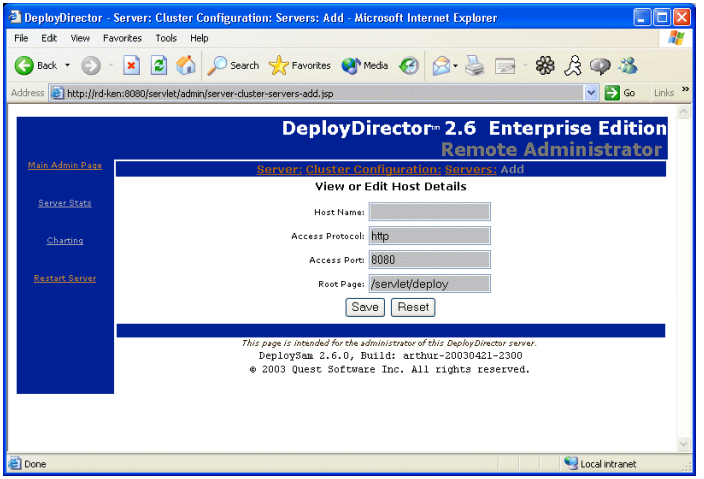
Defining server names at the cluster level

1. Ensure that your cluster of servers has been properly defined.

   Regardless of which servers are accessible by client-side end users, all servers that are meant to share logs and bundle changes must be part of the cluster.

2. In the Remote Administrator, navigate to the Server: Cluster Configuration: Server Names page.

3. Click Add Server to begin entering the new server details.



4. In the Host Name text field, enter the name by which the server will be known to client-side users. (Please note that this name may be different from its cluster host name.)

5. In the Access Protocol text field, enter the protocol used to access the new server (the default value is `http`).

6. In the Access Port text field, enter the port through which the server is accessed (the default is `8080`).

7. In the Root Page text field, enter the path through which the server's bundles can be accessed from the client side (the default is `/servlet/deploy`).

   The values entered as the machine and page properties constitute the base URL at which the server's bundles are accessed.

   For example, if you entered `installserver` in the Host Name field, and `http`, `8080`, and `/servlet/deploy` in the remaining fields, then the server's access URL is `http://installserver:8080/servlet/deploy`.

8. Review your settings, then click Save.

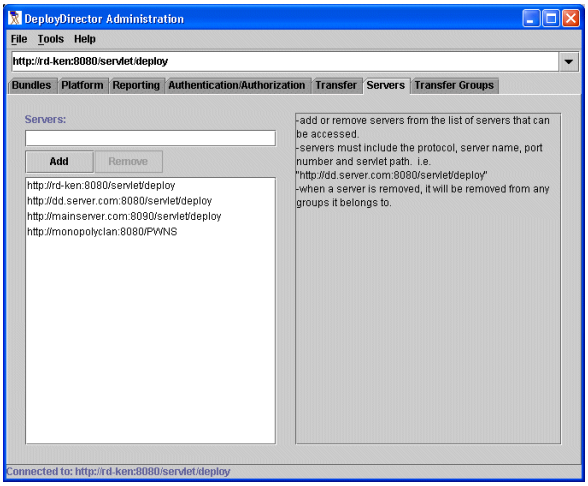
Transfer Groups

While the automatic replication of bundles within a cluster can ensure identically stocked server vaults, it is sometimes necessary to move bundles between clusters (e.g. from testing to deployment) or to upload bundles to multiple clusters (e.g. your network is geographically compartmentalized). You can use the Administration Tool to send bundles to established transfer groups. Setting up and using transfer groups allows you to:

- view the vault contents (i.e. bundles) of different servers to confirm that their vault contents are the same
- manually transfer new bundles to servers that are not part of a cluster
- move (copy) bundles between clusters (e.g. from a development cluster to a production cluster).

Listing Servers in the Administration Tool

The Administration Tool contains its own list of servers to which you can connect and transfer bundles. These servers can be both test servers and production servers, thus do not have to be a part of any cluster. You can view the list of servers known to the Administration Tool by clicking the Servers tab.



The same list of servers is found in the server combo box on the main tool bar. The selected server is the one to which the Administration Tool is currently connected, thus selecting another from the list results in an automatic connection attempt. (You will be prompted for proper authentication information.)

The removal of a server from the list is a matter of selecting one from the server list and clicking Remove. Adding a server requires that you enter its full access path.

Adding a new server to the servers list

1. Select the Servers tab in the Administration Tool to view the current list of known servers.
2. In the Servers text field, enter an access path for the server you wish to add to the list.

   The server path must include protocol, server name, port number, and servlet path information (e.g. `http://your.server.com:8080/servlet/deploy`).
3. Click Add.

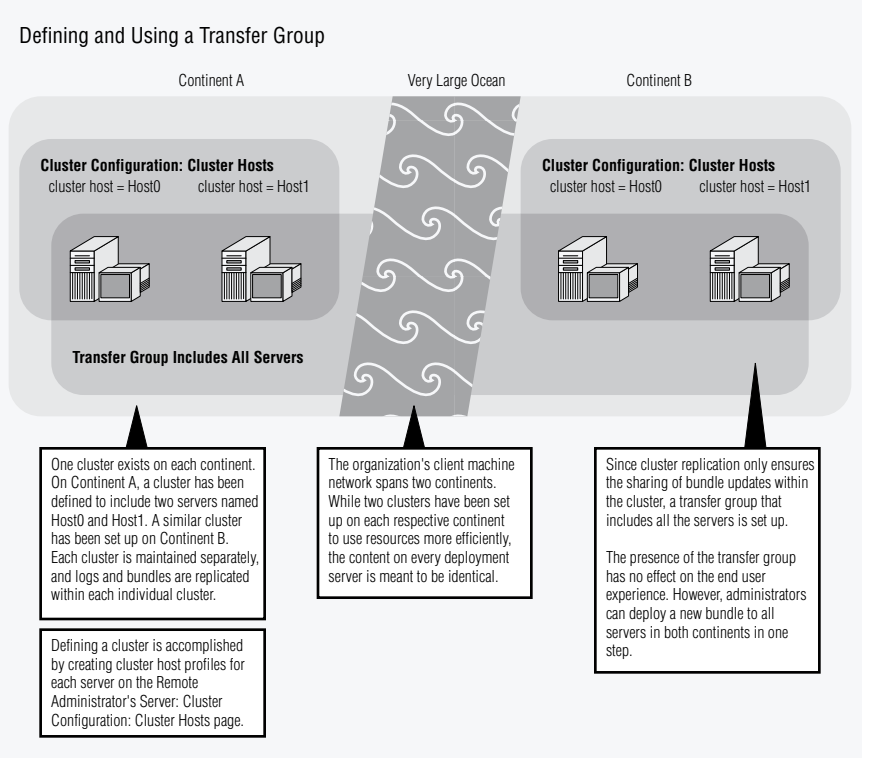
   The new server appears in the list.
4. Test the new server's access information by selecting it from the server combo box in the main tool bar.

   A successful connection attempt means the inputted server information is correct. If the connection attempt fails, check to see that it is currently running, and verify that the access path is correct.

Using the Servers List to Compile Transfer Groups

Once various deployment servers are set up in the Administration Tool, you can create transfer groups. A transfer group can be comprised of test servers or production servers, allowing bundle uploads to servers during a testing phase, and quick rollout to actual production servers. Typically, an administrator will connect to a test server on which new bundles have been created with the Administration Tool, then send out the new bundle to a transfer group.

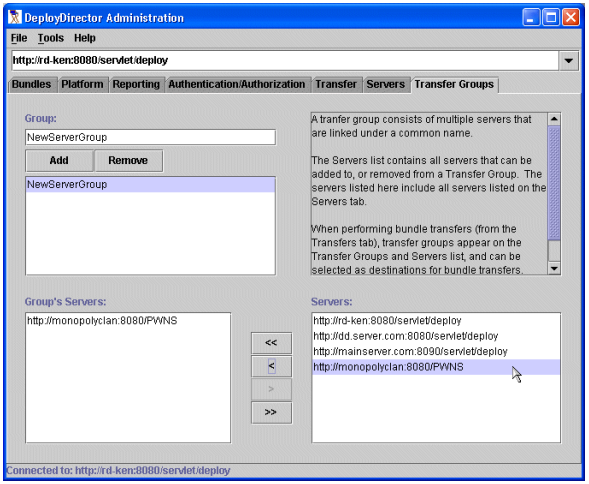
It is also possible to create transfer groups that are comprised of servers that make up a cluster. In the following example, a transfer group has been defined to include all the servers found in two clusters, allowing fast server updates across a geographically large deployment network.



Creating a transfer group

1. Ensure that all servers meant to be part of the transfer group have been added to the Administration Tool's servers list
2. In the Administration Tool, click the Transfer Groups tab.

   Here you will be able to name a transfer group and add servers from the known servers list.



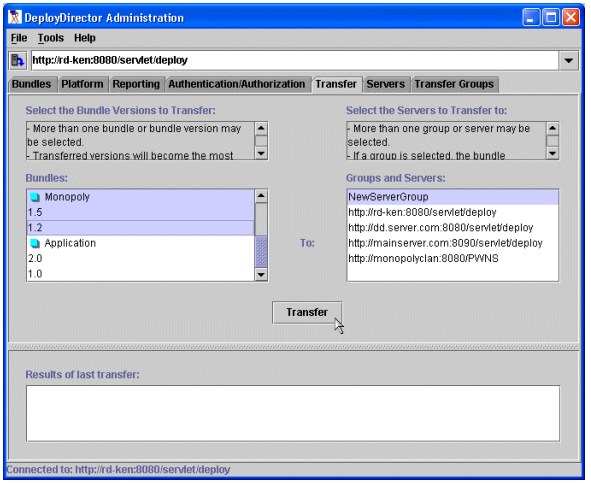
3. In the Group text field, enter the name for the transfer group.

   The transfer group is added to the group list, and is automatically selected.
4. From the Servers list, select the servers that are meant to be part of the transfer group.
5. Click the < key to make the selected servers part of the transfer group.

   Your transfer group has been created. You can now transfer bundles to all the servers in this group in one step.

Uploading bundles from a server to a transfer group

1. Ensure that a transfer group has been set up.
2. In the Administration Tool, click the Transfer tab.



   All bundles found on the server to which you are connected are displayed in the Bundles list. Each bundle's existing version names (i.e. versions that exist on this particular server) are listed beneath their respective parent bundle.

   All transfer groups, as well as all servers known to the Administration Tool are found in the Groups and Servers list.

3. In the Groups and Servers list, select the transfer groups and any other servers to which you want to transfer bundles.

   If an individually selected server is also found in a selected transfer group, bundles will only be transferred to that server once.

4. In the Bundles list, select the bundles or bundle versions that are going to be transferred to the selected transfer groups and servers.

   Selecting a bundle name selects its most recent version for transfer, and selecting multiple versions of the same bundle retains the version order when transferred to the destination server.

   If the bundle on the destination servers already exist, transferred bundles become the most recent versions on the destination server.

5. Click Transfer to upload the selected bundles to the transfer groups and servers.

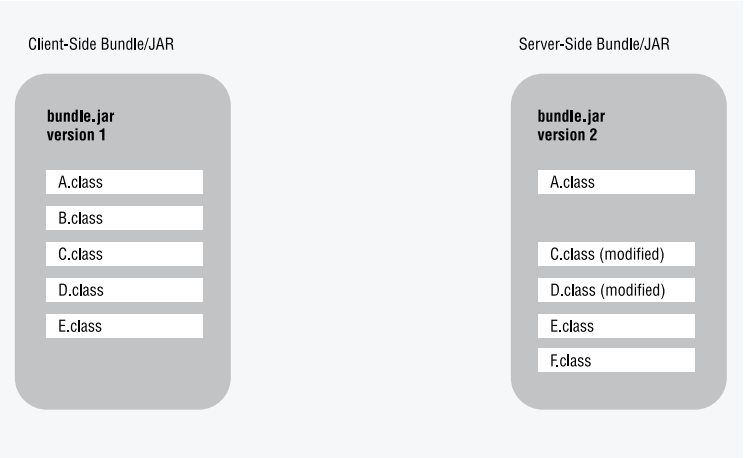
The Automatic Creation of Bundle Updates

When an end user requests a new bundle version, the SAM does not send that entire bundle to them. Instead, the SAM performs JAR differencing to create a smaller bundle version update, which is then sent to the client side for further processing. This technique cuts down on bandwidth use across the network and is particularly helpful when large groups of end users are simultaneously requesting and downloading a new bundle version.

Understanding JAR Differencing

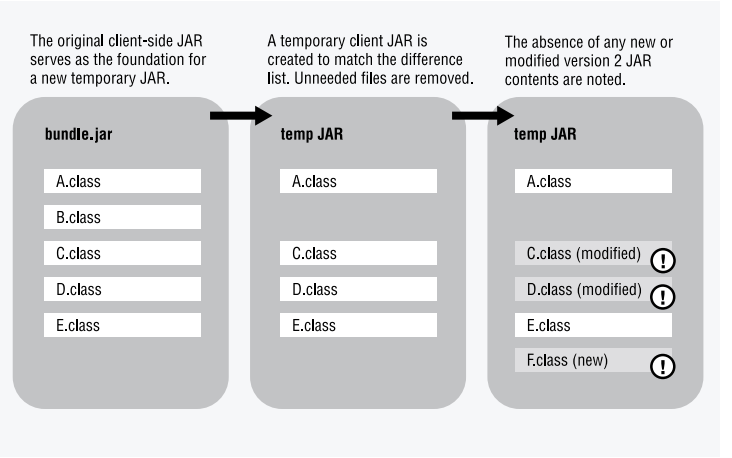
JAR differencing results in the creation of a temporary JAR. This update JAR consists only of the differences between what is in the new server-side version, and what already exists in the old client-side version. Once sent to the client side, this server-side JAR is combined with a modified client JAR to form the actual next version bundle.

To illustrate this process in action, consider these two JARs:

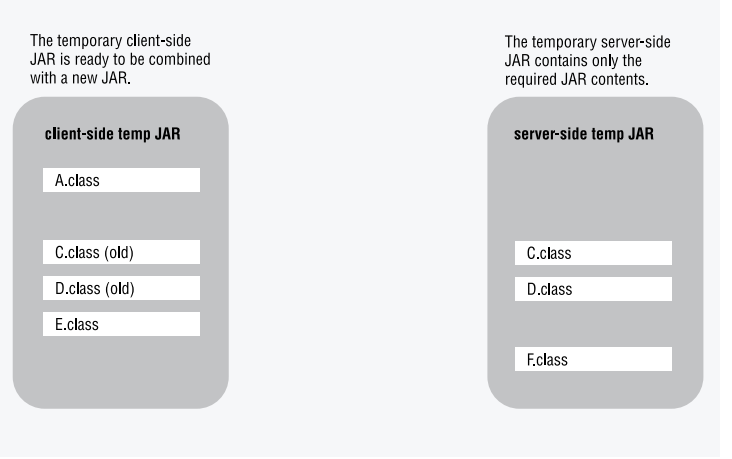


1. The JAR that constituted the first version of the bundle has been deployed to, and currently exists on, a client machine. The second version of the bundle, which currently resides on the deployment server, is being requested by the client.
2. The contents of both JARs are compared on the server side. In this example, the SAM determines that version 2 of the JAR no longer contains `B.class`, has a new class (`F.class`), and also has two modified classes (`C.class` and `D.class`).

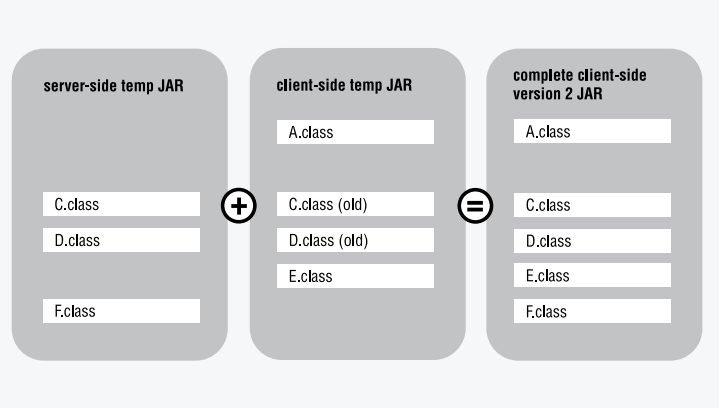
3. A list of differences is generated and sent to the client side, and is used by the CAM to begin creating a temporary JAR. If necessary, unneeded JAR contents from the current client-side version are removed, and a request is made for any other required JAR contents that are missing.



4. On the server side, the JAR content difference list is used to create a server-side temporary JAR that will be sent to the client side. This JAR contains anything that the client-side JAR does not have.



5. The server-side temporary JAR is sent to the client side. The CAM rebuilds the final version 2 JAR by combining the contents of the received server-side JAR, and the modified client JAR.



Server Caching

The offsetting aspect of building bundle updates on the fly is the server-side resource overhead that may occur, depending on the hardware capabilities of the deployment servers on which SAMs are found. The server-side cache, which temporarily stores these updates (as ZIP files), alleviates heavy disk usage during these times.

If the server-side cache is enabled, it is always checked first for the bundle update an end user is requesting. By default, the maximum cache size is set to 30 million bytes (~28MB) and its contents are kept for 30 days. The contents of the cache can be found at:

`<installpath>/deploydirector/dd/cache`.

Setting server-side cache properties

1. In the Remote Administrator, navigate to the Server: Server Configuration: Miscellaneous Properties page.
2. Enter a value in the deploy.server.cache.maxage field.

   This value should be an explicitly stated time interval (e.g. `30 days`). This indicates for how long any cache items can sit before being removed by the SAM. When a cache item is used, its counter is reset.

   Please refer to the Administration Tool Date and Time Entry Formats section in Chapter 2, Introduction for more information about valid time and date formats that can be used with the Administration Tool.
3. Enter a byte value in the deploy.server.cache.size field.
4. Click Update Configuration to commit these changes to the server.

Since hardware capabilities vary from server to server, this property must be set at the server level and must be individually set for all servers that are part of your deployment network.

Running DeployDirector as a Windows Service

When you first installed DeployDirector, you installed and set up the standalone server (you can revisit Configuring and Running the Standalone Server in Chapter 1 to refresh your memory). This is the prepackaged server on which DeployDirector can run, as an alternative to using it with a commercial application server.

If you are using Windows 2000 or Windows NT, you can run the standalone server as a Windows service. As a Windows service, DeployDirector can be run transparently in the background without any user interface. It automatically starts and stops when the machine is turned on and off, is not tied to any specific administrator's user ID and password, and remains active whether or not anyone is logged in to the machine on which it runs.

To run DeployDirector as a service, locate the executable `service.bat` file in the `<ddinstalldirectory>/standalone/bin` directory. At a command prompt, use this command with the `install`, or `uninstall` parameter to set up, or remove DeployDirector as a server. If you enter:

```
service install
```

you will be able to see it in your Windows Services Control Panel.

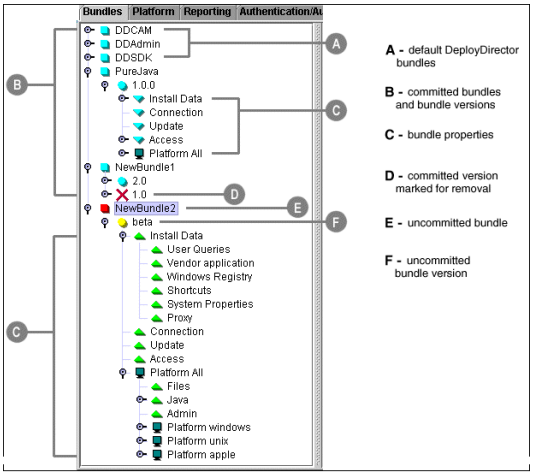
When running as a server, any error reporting or standard output will be recorded to the `stderr.log` and `stdout.log` files, which are found in the `<ddinstalldirectory>/standalone/logs` directory.

Chapter 4

Adding Bundles and Defining Bundle Content

The core of every bundle consists an application your organization has developed. Before configuring a bundle's deployment properties, you first define its content. Whether the application is a straightforward update for a single platform, or is a large application for use on different client platforms, the Administration Tool assists in the assembling of application files and directory structures into a server-bound bundle.

One of the more common tasks performed with the Administration Tool is changing the contents of the vault (i.e. bundles and bundle versions). In the Bundles tab, the contents of the vault on the server to which you are currently connected are displayed:



Making Changes to the Vault

From the Bundles tab, you can add or remove bundles, as well as create new bundle versions, either from scratch, or by basing it on an existing bundle version. Bundles based on existing versions already contain files that you can then modify. Otherwise, newly created bundles require that you define its internal directory structure and accompanying files.

Adding and Removing Bundles

Important: When naming bundle versions, be aware of the authorization class with which it works. Some authorization classes introduce constraints on allowable version name formats. Please refer to Authorization Behavior and Allowable Bundle Version Names in Chapter 9, for more information.

While the addition and removal of bundles to your deployment server is a straightforward process, it is important that bundles and bundle versions are logically named. Two conventions that come from this logic include the naming of bundle versions in an incremental manner, as well as bundle and bundle version names being unique, particularly when a bundle version rollback occurs.

Adding new bundles to the vault

1. Click anywhere in the node list to bring focus to it.
2. Click Edit > New Bundle.

   The New Bundle dialog appears, prompting you for the new bundle's name.
3. Enter a name for the bundle.

   The New Bundle Version dialog appears. When you create a new bundle, you are also required to create an initial bundle version.
4. Enter a name for the initial version (i.e. a name or number).

   The new bundle appears as a top-level entry in the node list as an uncommitted bundle (indicated by the red marker), and the initial version is displayed as a child node, also as uncommitted (indicated by the yellow marker).

   Some of the initial bundle version's default properties are set, but you will need to configure it for deployment by adding files and setting deployment and installation properties.
5. Configure the bundle.

   Information on adding files to bundles is discussed later in this Administrator's Guide chapter. Subsequent chapters provide information about all other bundle properties that need to be configured.

6. Click File > Update Server to commit your new bundle and initial version to the server once it has been configured.

Adding new bundle versions

1. Select the bundle for which you want to add a new version.
2. Click Edit > New Version.

   The New Version dialog appears.
3. Enter a name for the new version.

   The new bundle version appears as an uncommitted child node (indicated by its yellow marker).

   Some of the initial bundle version's default properties are set, but you will need to configure it for deployment by adding files and setting deployment and installation properties.
4. Configure the bundle version.
5. Click File > Update Server to commit your new bundle version to the server.

Removing bundles

1. Select the bundle or bundle version you want to remove.
2. Click Edit > Remove.

   If the bundle has not yet been uploaded to the server, it is immediately removed from the list.

   If the bundle was previously uploaded to the server, it is marked for removal the next time you update the server with your changes.
3. Click File > Update Server to commit this change (i.e. remove the bundle from the server).

Rolling back bundle versions

1. Select the bundle version you want to remove.
2. Click Edit > Remove.
3. Click File > Update Server to remove the bundle from the server.
4. Ensure the next bundle version you create to replace the rolled back version is not identically named (e.g. if version 2.0 of a bundle is rolled back, its replacement version could be named '2.1' or '2.0a').

Basing New Bundles on Existing Bundles

It is not uncommon for an organization to periodically develop application updates that need to be deployed to clients. When adding files to, and setting properties for a new bundle version, it is not necessary to build it from scratch every time. DeployDirector allows the creation of a new bundle version based on the settings and contents of the previous version. The Administration Tool offers two options for basing new bundles on existing bundles: copying bundle versions from a deployment server, and copying bundle versions from a local source drive.

Copying a bundle version that has already been uploaded to a server results in a new bundle version that has inherited all the property settings and files of that server-based bundle. Once created, you can then modify the bundle and upload it to the server as a new version.

Copying a bundle version from a defined, local source allows administrators to retrieve only the latest application changes directly from its source drive on the network. Typically, when you are adding files to a newly created bundle, these files are most likely coming from a work area on your local drive, or a development drive on the network. If any of these source files have since been modified, it is more desirable that a new bundle version automatically retrieves these newer local files, rather than you having to manually do this. Use the Administration Tool's Copy Source Version option to accomplish this.

Copying bundle versions from the server

1. Select the committed bundle version that you want to copy.
2. Click Edit > Copy Server Version.

   The Copy Server Version dialog appears.
3. Enter a name for the new version.

   The copied bundle appears as an uncommitted child node in the bundle list.
4. If necessary, configure the bundle version by modifying its properties and adding or removing files that are to be deployed.
5. Click File > Update Server to commit your new bundle to the server.

Copying bundle versions from the local source

1. Select the server-based bundle version that you want to copy.
2. Click Edit > Copy Source Version.

   The Copy Source Version dialog appears.
3. Enter a name for the new version.

The source files are copied to your bundle. The procedure works on the assumption that all files still exist in the locations from which they were originally retrieved.

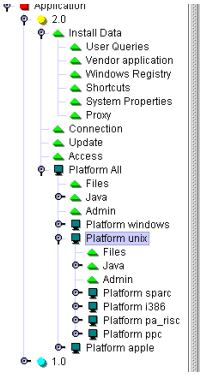
The copied bundle appears as an uncommitted child node in the bundle list.

4. If necessary, configure the bundle version by modifying its properties and adding or removing files that are to be deployed.
5. Click File > Update Server to commit your new bundle to the server.

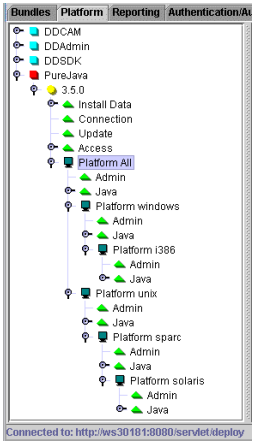
Adding Files and Directories to Bundles

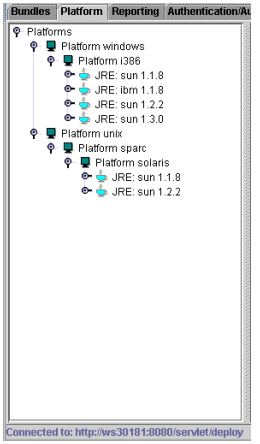
Bundle contents begin as files found on a system administrator's local drive or on a network. When the files are ready to be deployed to end users, a new bundle version is created (from scratch, or copied, as discussed in the previous section), and the attributes associated with the bundle can be configured and re-configured until it is uploaded and committed to the server.

You can add files to a bundle when a Platform node is selected in the Administration Tool. Selecting a particular Platform node indicates that any files included when that node is selected are meant to be deployed to clients based on that particular platform. For example, files found and added while the Platform All node is selected are meant to be deployed to all clients on any platform, while files added when the Platform Unix node is selected will only be deployed to Unix clients.



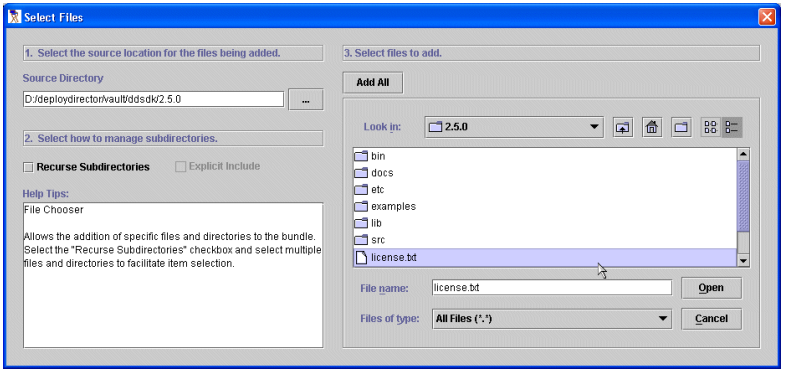
Additionally, under each main platform node (i.e. Windows and Unix), specific platform nodes can exist (see image below left). The list of available platforms in a bundle's list is dependent on which JREs are present on your deployment server, which can be viewed in the Platform tab (below right).





When adding files to a new bundle, it is recommended that you add common files under the All platform first, then add specific files meant for the supported platforms by selecting the desired Platform node and adding files.

When adding files to a bundle, the file selection dialog that appears prompts you to set the source directory. This is considered the root directory relative to your local file system, on which the files that are added are based (and affects the `SOURCE` tag in the bundle's `version.xml` file). The right portion of the dialog allows you to choose the files and folders you want to add to the bundle.



When selecting bundle contents, the selection of a folder includes all of its contents (including nested folders). Avoiding a comprehensive inclusion of folder contents can be done by manually selecting files within. The paths and location of the files, relative to the source directory, are retained.

In addition to selecting and including entire directory-file structures, you can also add individual folders. You cannot add files to these folders when creating a bundle; their inclusion results in the creation of an empty folder on the client side after installation. Add folders this way if you want your installed bundle to create empty directories for future use (e.g. creating an empty documentation folder with an application bundle acts as a placeholder for documentation files that can be downloaded in a separate documentation bundle).

Adding individual directories to a bundle's file structure

1. Select the bundle version's appropriate Platform node.
2. Click Edit > Add Folder.
3. In the Add Folder dialog, enter the name of the folder you would like to create.

   Upon installation, this name will be appended to the bundle's installation path (i.e. `[installdir]/[vendorname]/[bundlename]/[addedfolder]`).

   The value entered should *not* contain drive letters or colons (e.g. `c:\temp`).
4. Continue to add files and folders to the bundle under this or other platforms.
5. Click File > Update Server to commit your new bundle to the server.

Adding cross-platform files

1. Select the bundle version's Platform All node.

   Once this node has been selected, you are able to add files and folders to the bundle, as indicated by the Edit menu.
2. Click Edit > Add Files.

   The Select Files dialog appears.
3. In the Source Directory field, enter or browse to the directory you would like to set as the source.
4. Select the files.
5. Click Open.

   The files you selected now appear under the bundle's Platform All node.

6. If required, add any platform-specific (i.e. Windows or Unix) files to the bundle.
7. Continue to configure the bundle version by modifying its properties and adding or removing files that are to be deployed.
8. Click File > Update Server to commit your new bundle to the server.

Adding Windows files

1. Select the bundle version's Platform Windows node, or the specific Windows platform child node.

   Once this node has been selected, you are able to add Windows files to the bundle, as indicated by the Edit menu.

2. Click Edit > Add Files.

   The Select Files dialog appears.

3. In the Source Directory field, enter or browse to the directory you would like to set as the source.
4. Select the Windows files.
5. Click Open.

   The files you selected now appear under the bundle's Platform Windows node.

6. If required, add any universal or Unix files to the bundle.
7. Continue to configure the bundle version by modifying its properties and adding or removing files that are to be deployed.
8. Click File > Update Server to commit your new bundle to the server.

Adding Unix files

1. Select the bundle version's Platform Unix node, or the specific Unix platform child node.

   Once this node has been selected, you are able to add Unix files to the bundle, as indicated by the Edit menu.

2. Click Edit > Add Files.

   The Select Files dialog appears.

3. In the Source Directory field, enter or browse to the directory you would like to set as the source.
4. Select the Unix files.
5. Click Open.

The files you selected now appear under the bundle's Platform Unix node.

6. If required, add any universal or Windows files to the bundle.
7. Continue to configure the bundle version by modifying its properties and adding or removing files that are to be deployed.
8. Click File > Update Server to commit your new bundle to the server.

Removing files

1. Expand the uncommitted bundle version's Platform nodes to reveal the files that it currently contains.
2. Select the files you want to remove.
3. Click Edit > Remove.

   The selected files are removed from the list and this bundle version.

   Continue to make changes to the bundle, then click File > Update Server to commit these changes to the server.

Chapter 5
Configuring Bundle Installation Properties

When a bundle's contents have been defined (as discussed in the previous chapter), the next stage in its configuration is to set its installation properties. While DeployDirector provides features that allow the creation of bundle installation CDs, its greatest strength lies in the deployment of bundles and updates over a network. Deploying bundles in this manner means client-side users will install bundles with a DeployDirector install applet in their Web browsers. As such, you need to configure all bundle properties that relate to its installation on the client side.

The Deployment of Bundles Via Web Browsers

When client-side users request a bundle, this request is initially executed and processed by the installer applet. This applet works in tandem with the CAM to simplify application installation on client-side machines. The absence of direct manual installation from the deployment process removes a mutual burden from both system administrators and client-side end users. (There are cases when a manual installation from a CD is warranted. Please refer to Preparing Bundles for Manual CD Installations in Chapter 8 for more information.)

Introducing the Installer Applet

The installer applet is a core component between the client-side end user and the SAMs. It shares duties with the CAM and is primarily responsible for installing the CAM on the client machine. The installer applet is the first component that client machines will execute during a bundle installation request. Essentially, it:

- shows the license page if one exists

- performs the client authentication/authorization check
- installs the CAM if one does not exist on the client machine
- installs a JRE to run the CAM if required
- downloads and installs the JRE required by the bundle
- asks Windows users if they would like a desktop shortcut to the application to be created (if this option has been enabled)
- runs the CAM and tells it which application to install (including the creation of shortcuts and executable files)
- displays the readme if one exists
- calls the launcher applet to start the application if the bundle was requested to be launched (please see The /launch Request later in this section for more information).

Re-Signing the Installer and Launcher Applets

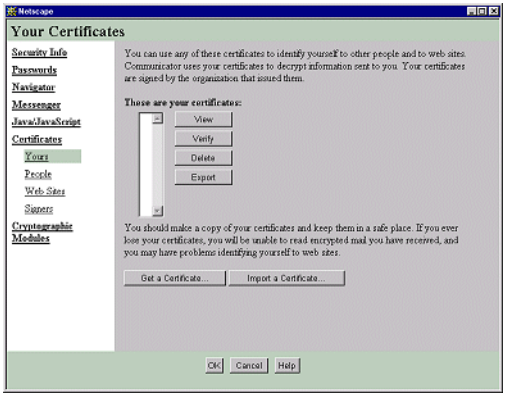
The installer applet runs in a browser, thus must be digitally signed in order to have access to system resources on the client side. The applet must be granted permissions to have network access, full file system access (read/write), system property read/write access, and to execute sub-processes. A signed installer applet is provided for you, however DeployDirector allows you to re-sign the installer applet with your own certificate.

The tools and certificates for re-signing the applet differ depending on the browser. The procedures outlined below have been tested with DeployDirector on the Windows platform. For Netscape Communicator, you can use the Netscape Signing Tool to sign the `install.jar` file located in the `<installpath>/deploydirector/installer` directory. For Internet Explorer, you will need the Microsoft SDK for Java which contains `signcode.exe` file used to sign the `install.cab` located in the `<installpath>/deploydirector/installer` directory.

Re-signing the installer applet for Netscape

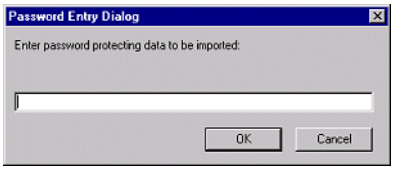
1. Download the Netscape Signing Tool for your platform from Netscape's Developer site http://developer.netscape.com.
2. Set your PATH variable to include `signtool.exe`.
3. Obtain a certificate (https://certs.netscape.com/). The certificate should be PKCS#12 (*.p12).
4. In Netscape Communicator, select the Security icon.

5. In the dialog box shown below, select Yours under Certificates and click the Import a Certificate button.



6. In the File Name to Import dialog, navigate to the certificate and click Open.

7. In the Password Entry Dialog, enter your privacy protection password. This password was created when you obtained the certificate.



8. Exit Netscape.

   The Netscape `cert7.db` file, located in the `Netscape/Users` directory, has now been updated.

9. Extract the install.jar from `<installpath>/deploydirector/installer` into a local directory.

10. Run the DOS shell or any command line shell.

11. Navigate to the directory to which you have extracted the `install.jar`.

12. Enter the following command:

```
signtool -d"[directory containing the cert7.db file]" -
z"install.jar" -k"[certificate name]" -p"[certificate password]"
[directory to which you extracted the install.jar]
```

    Make sure that each argument for the signtool is contained within quotation marks, as shown above, if it has any spaces or other non-alphanumeric characters.

Re-signing the installer applet for Internet Explorer

1. Download the Microsoft SDK for Java from http://www.microsoft.com/java/download.htm#32.

   The CAB file has to be signed at a low level, in order to allow it to run in the "low" security zone on the client desktop. You may find it helpful to consult the Developer FAQ at http://www.microsoft.com/java/security/secfaq.htm.

2. Ensure that signcode.exe is in the PATH.

3. Obtain a certificate (https://certs.netscape.com/).

   You should obtain `mycert.spc` and `mykey.pvk` files. For more information, please consult http://www.thawte.com/certs/developer/msauthenticode.html.

4. Run the DOS shell.

5. Enter the following command:

   ```
   signcode -j "[directory path to javasign.dll]" -jp low -spc
   "[directory path to mycert.spc]" -v "[directory path to
   mykey.pvk]" [directory path to the install.cab]
   ```

   Ensure that each directory path in this command contains the indicated target file.

Launching Applications

DeployDirector applications can be launched by the end user by clicking on the desktop shortcut, if one was created during bundle configuration in the Administration Tool, or by using the `/launch` request in the Web browser. Launching applications through a Web browser gives administrators an ability to pass parameters to the application.

The /launch Request

The `/launch` request calls the launcher applet which automatically runs an application that is already installed on the client machine (only Windows clients are currently supported). If the application has not been installed when the request is made, the launcher applet redirects to the installer applet.

The separate installer and launcher functionality ensure that only the required streamlined applet is downloaded during a deployment session. If the application is detected on the client side, then only the launcher applet is downloaded, saving bandwidth and decreasing the download time.

Having end users run applications with the `/launch` request may be preferred if your organization wishes to ensure that they are run identically by all users. This may be necessitated by the presence of a large number of users, or the existence of policies that require permission for new desktop shortcuts to be created on client machines. In a case like this, bundles can be configured to not install desktop shortcuts, and client-side end users can use a department or organization-wide launch page (with hypertext links) to run applications using the `/launch` request.

If you plan on running bundles with the `/launch` request, it is required that you configure your bundles so that the User Queries Install Directory option is disabled (please see Bundle Installation Directories: Creation Strategies), and set the system property `user.dir` to `$(INSTALLDIR)` (please see Determining how Bundles Affect Client Machine Settings for more information about system properties). Using this Java system property ensures the current/target directory is always the bundle's install directory (i.e. [`installdir`]/[`vendorname`]/[`bundlename`]). Since browsers that work with the launch applet, unlike desktop shortcuts and startup scripts, have no hard-coded information about what the bundle's current directory information should be, they could be pointing off into space. Ensuring focus by setting the `user.dir` system property rectifies this problem.

Customizing the Install, Launch, and Error Pages

DeployDirector includes a standard HTML page in which the installer applet is run. This page was first seen when the Administration Tool was deployed to a system administrator's workstation, and appears when a proper URL is entered in a Web browser either manually or through a hypertext link.

DeployDirector provides you with an option of customizing the user interface of the install and launch pages displayed on the client side. These pages are a mixture of static and dynamic content. The static content for the install and launch pages is read from the `application.html` file, located in the server's `deploydirector/etc` directory.

Before the page is loaded, it is scanned for place holders. These are special comments in the HTML which are replaced with dynamic content as the page is loaded. The `application.html` page can be globally edited for all applications or for each version of a bundle. The only requirement is that the tag
`<!--PlaceHolder:APPLET-->` is included somewhere in that HTML page. This tag is replaced by the installer applet or the launcher applet, as required.

The following is a list of tags that can be included in your custom install/launch HTML page. These tags can appear anywhere in the HTML page and there can be multiple instances of any tag.

Tag	Replaced with:
<!--PlaceHolder:LAUNCHORINSTALL-->	“install” or “launch”, as requested by the end user
<!--PlaceHolder:APPLICATION-->	The name of the application (read from the bundle’s version.xml file).
<!--PlaceHolder:VERSION-->	The version of the application (read from the bundle’s version.xml file).
<!--PlaceHolder:URL-->	URL entered by the user into the browser to get to the install or launch page.
<!--PlaceHolder:LOGO--> <!--PlaceHolder:LOGO_SS-->	Default Quest Software DeployDirector logo or custom logo found in the deploydirector\etc directory. The custom logo should be saved with the same file name.
<!--PlaceHolder:APPLET-->	The installer applet or the launcher applet, as required. This tag is REQUIRED.
<!--PlaceHolder:SERVLET NAME-->	The name of the servlet read from the server.properties file. If the property deploy.server.name is not provided, the default “DeploySAM” is used.

If no custom page is provided, then the `application.html` template found in the `deploydirector\etc` directory is used and the tags in the HTML page are replaced with the corresponding values.

Customizing the install and launch pages

1. Create a custom HTML page based on the `application.html` file found in the `deploydirector/etc` directory. Make sure that it includes the `<!--PlaceHolder:APPLET-->` tag.
2. Using the Administration Tool, configure the bundle and commit it to the server.

   Ensure that the bundle version is committed to the server before adding the customized install/launch page.
3. Click on the Set Install/Launch Page button on the `bundle_name/bundle_version` node.

4. Navigate to the directory that contains your custom install/launch HTML page and click Open.

   Note that your custom HTML page does not need to reside in a specific location. It also does not need to be called `application.html`, although the Administration Tool renames it to `application.html` before sending the page to the server. Your customized HTML page is added to the following directory:
   `deploydirector/vault/<bundle_name>/<bundle_version>/dd/application.html`

   The next time the end user will attempt to install or launch the deployed application, the customized install launch page will be displayed on the client side.

The Error Page

DeployDirector server notifies the end user of any errors by displaying an error page. This page is composed of static and dynamic content. The static content is read from the `error.html` file found in the server's `deploydirector/etc` directory. This page contains place holder HTML tags which are replaced with the dynamic content as the page is loaded. You can customize the error page by using DeployDirector's custom HTML tags.

The following is a list of the custom HTML tags supported by the error page. These tags can appear anywhere in the HTML page and there can be multiple instances of any tag.

Tag	Replaced with:
<!--PlaceHolder:SERVLET NAME-->	The name of the servlet read from the server.properties file. If the property deploy.server.name is not provided, the default "DeploySAM" is used.
<!--PlaceHolder:ERROR TEXT-->	The error message displayed by the server.
<!--PlaceHolder:REQUEST-->	The request from the user which generated the error.
<!--PlaceHolder:URL-->	Full URL to the page that caused the error.
<!--PlaceHolder:LOGO--> <!--PlaceHolder:LOGO_SS-->	Default Quest Software DeployDirector logo or custom logo found in the deploydirector\etc directory. The custom logo should be saved with the same file name (deploydirector.gif or deploydirector_ss.gif).

Passing URL Parameters to an Application

Whenever a DeployDirector-administered application is launched through a browser using the `/launch` request, additional parameters can be passed to that application. This functionality offers more control over how applications are run.

If the application already contains hard coded arguments, any additional parameters passed through the URL will be appended. For example, suppose that arguments a, b, and c, are hard coded in the bundle, and the end user enters the following URL in the browser:

```
http://localhost/servlet/PureJava/launch?d&e&f=1
```

The application will receive arguments a, b, c, d, e, and f=1.

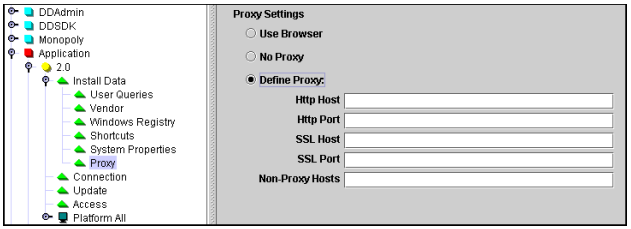
For information on setting hard coded (or execution) arguments, see the Defining Entry Points in Chapter 6. Any additional parameters can be passed through the URL.

Configuring Proxy Settings

In a many organizations, client machines access deployment servers through a network proxy. In this environment, DeployDirector bundles must contain proxy information to ensure communication between the client and server machines (e.g. for initial deployment, bundle updates). Organizations may also use an SSL proxy in conjunction with, or instead of a network proxy.

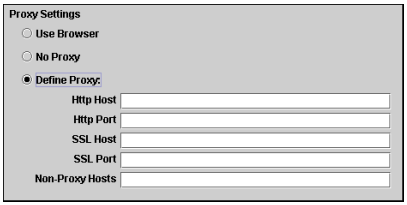
Whether and how proxy information is handled with bundles depends on how they are configured in the Administration Tool. A bundle can be configured to contain proxy information before it is deployed, request the client-side user to enter the information during deployment, or a combination of the two. Properties related to proxy configuration are found when the Proxy and User Queries property nodes have been selected in the Administration Tool.

Selecting the Proxy node reveals three possible configuration options for the bundle: Use Browser, No Proxy, and Define Proxy.



- The Use Browser setting configures the bundle to detect and use the proxy settings of the client-side browser (whether it is Internet Explorer or Netscape Navigator) during the initial installation.
- Define Proxy configures the bundle to use the proxy information that is entered in the accompanying text fields. This option can be used if the proxy server for deployment is different from the one used for non-deployment tasks, or if there is an SSL proxy.
- No Proxy prevents the use of a proxy even if the client's browser is configured to use one.

While the options under the Proxy node configure a bundle's proxy settings before deployment, selecting the User Queries node reveals the CAM Config check box, which enables or disables proxy configuration *during* bundle deployment. If it is enabled, the client-side user will be shown, as part of the bundle's installation process, proxy configuration fields.



In the above example, note that the HTTP Proxy Host text field is empty. If a client-side end user was presented with this screen during the installation of a bundle, they would have the opportunity to enter the proxy information on their own. But, since the field is initially empty, it indicates that no default proxy information was configured in the bundle.

This situation represents one of six possible outcomes when configured properties under both property nodes are used together. It is important to be aware and take advantage of the results of different combinations. The following table outlines the outcomes on the client side with different configurations under the Proxy node and User Queries node.

Proxy Node, Proxy Setting	User Queries Node, CAM Config	Outcome during bundle installation
Use Browser	enabled	client browser proxy information used and shown, but user asked to confirm or input them
Use Browser	disabled	**(default setting for new bundle versions)** client browser settings used but not shown, and user not asked to confirm or input them
No Proxy	enabled	no proxy information used, thus no default information shown; user asked to input proxy information
No Proxy	disabled	no proxy information used; nothing shown to user
Define Proxy	enabled	proxy information set by system administrator; user asked to confirm or edit settings
Define Proxy	disabled	proxy information set by system administrator; user does not see them

When setting proxy information, it is important to remember that the CAM only operates through one proxy, and will always use the proxy information from the last bundle it installed or updated. If proxy settings change, this must be reflected in all affected bundles (i.e. new bundle versions with updated proxy information must be created). However, it is equally important to keep the old proxy in existence long enough for all users to download the new bundle versions.

Configuring Browsers to Use Proxy Information

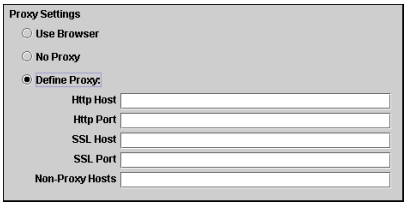
In order for client-side users with Netscape Navigator to receive proxy information, JavaScript must be enabled in their browsers. Additionally, client-side users with Microsoft Internet Explorer who disable proxy settings in their browsers after a connection has been made will need to restart their browser or clear their browser cache in order for the new settings to function. This behavior is exclusive to Internet Explorer, as it retains proxy information for visited URLs even after the proxy has been disabled.

Deploying with Proxies Present on the Network

When a proxy is present on your network, if your deployment network uses a cluster, in order for the CAM and SAM to communicate properly through the proxy, your cluster hosts need to be assigned server names. This is accomplished by setting server names on the Server: Cluster Configuration: Server Names page using the Remote Administrator. (Please refer to The Client-Side Visibility of Servers in a Cluster in Chapter 3.)

Configuring proxy information in the Administration Tool

1. Expand the Install Data node.
2. Select the Proxy node.
3. In the right pane, select the proxy setting that matches your needs for your deployment network.



   If Define Proxy is selected, enter the host name of the proxy, and the port on which it can be found. You can also enter similar information for an SSL proxy in place of, or in addition to the standard proxy.

   Finally, list any host names that can be reached without the use of the proxy (which is rare, but possible).

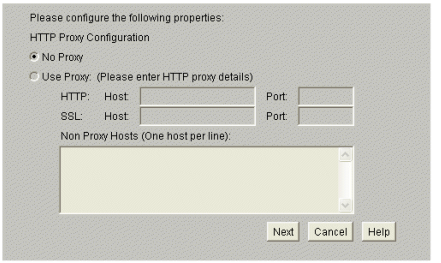
   The following are the guidelines for listing non-proxy host names:

   - Entries should be separated with "|"
   - Host names and IP addresses are valid, e.g. "`11.22.33.44|www.quest.com`"
   - In order to allow the matching of any host in a domain, start the host name with "." (e.g. entering "`.quest.com`" will match any host name that was a part of `*.quest.com`, such as `www.quest.com`, or `ftp.quest.com`).

4. Continue to set other bundle properties, then click File > Update to commit these changes.

Allowing client side users to configure proxy information

1. Expand the Install Data node.
2. Select the User Queries node.
3. In the right pane, select the CAM Configuration check box if during the bundle installation process you want the client-side user to be shown the proxy information used by the bundle, with the possibility of editing.



When the check box is enabled, the proxy configuration dialog is included as part of the bundle's installation process.

Depending on how you configure settings under the Proxy node, client-side end users may be required to enter proxy information (including the proxy host name and port and, if necessary, any non-proxy hosts). It is recommended that you provide this information on your main HTML-based installation page for client-side users if it is necessary for users to enter values themselves.

4. Continue to set other bundle properties, then click File > Update to commit this change.

Passing Cookies to the Installer or Launcher Applet

In typical Web environments, when clients connect to Web servers, they receive and store cookies. This information is used the next time the client establishes a connection with the server. By default, the introduction of DeployDirector to your network prevents the normal passing of cookies. However, DeployDirector can be configured to accommodate the use of cookies for identification (i.e. authentication and authorization) for single sessions.

Once properly set up, DeployDirector can pass a cookie to its installer or launcher applet which will then by used by the CAM (or the application from the end user's perspective) to connect to the server with identification information.

Configuring DeployDirector to Pass and Use Cookies

With typical client-Web server interaction, the client connects to a Web server which returns a cookie that remains on the client machine. When using DeployDirector, a cookie is sent to the launcher or installer applet and is saved on the client side. This cookie is then located and used by the CAM whenever client-server communication occurs (i.e. when the application that uses the cookie has started).

The implementation of this functionality is one that emphasizes automation. Defining cookie parameters in the access URL for the application is all that is required to enable cookie passing. Alternatively, you can customize the launcher or installer's front end HTML file yourself to include the cookie information. (Please see Customizing the Install, Launch, and Error Pages on page 81 for more information on customizing front end HTML pages.)

Enabling cookie passing to the installer or launcher applet automatically:

1. Ensure the application meant to be used with the cookie has been committed to the server.
2. Ensure the completed cookie file has been placed on the server, so that DeployDirector can retrieve it when it needs to pass it to the launcher or installer applet.
3. When creating an access URL for end users (e.g. on an HTML-based launch page for all of your applications), create the launch or install URL for the bundle to include cookie parameters so that a URL such as this:
   `http://localhost/servlet/application/launch`

   you would include cookie parameters:
   ```
   http://localhost/servlet/application/launch?COOKIE=
         name=cookiename;value=cookievalue;date=expirydate
   ```
4. For the mandatory `name` parameter, enter the name of the cookie.
5. For the mandatory `value` parameter, enter the string that represents the cookie
6. For the mandatory `date` parameter, enter a properly formatted date string that adheres to accepted specifications (i.e. RFC822, 850/1036, 1123, or Netscape cookie specifications). This date defines when the cookie will expire.
7. For the optional `domain` parameter, (whose default value is the host name of the server that originally sent the cookie), you can modify it to match another domain (e.g. "`.quest.com`").

   Once this cookie parameter has been added to the launch or install command, whenever a client-side user launches or installs the bundle, the cookie will be retrieved from the (which) directory, and sent to the client side to be locally stored.

Enabling cookie passing to the installer or launcher applet manually:

1. Ensure the application meant to be used with the cookie has been committed to the server.
2. Ensure the completed cookie file has been placed on the server, so that DeployDirector can retrieve it when it needs to pass it to the launcher or installer applet.

3. Create a customized `application.html` page for the launcher or installer applet, which will include the cookie applet call.

Important: This sample applet code is meant to server only as an example of how to set cookie parameters. Ensure you have entered settings for your own applet and cookie.

4. In the custom application.html file, replace the `<!--Placeholder:APPLET-->` comment with the following:

```
<APPLET>
     CODE="com/yourorganization/install/Installer.class"
     CODEBASE="."
     ARCHIVE="http://localhost:8080/install/
          <!--PlaceHolder:APPLICATION-->/install.jar"
     NAME="<!--PlaceHolder:APPLICATION-->, version
          <!--PlaceHolder:VERSION> Installer"
     WIDTH=500
     HEIGHT=300
     alt="You would need a Java-enabled browser to see this."
     <PARAM NAME="CABBASE" VALUE="http://localhost:8080/
          install/<!--PlaceHolder:APPLICATION-->/
          install.cab">
     <PARAM NAME="BUNDLE" VALUE=
          "<!--PlaceHolder:APPLICATION-->">
     <PARAM NAME="VERSION" VALUE=
          "<!--PlaceHolder:VERSION-->">
     <PARAM NAME="SERVER" VALUE="http://localhost:8080/
          install/bundle">
     <PARAM NAME="LAUNCH" VALUE="false">
     <PARAM NAME="NEXT_PAGE" VALUE="">
     <PARAM NAME="COOKIE"
          VALUE="name=cookiename;value=cookievalue;domain=
          .yourorganization.com;expiry=expirydate>
</APPLET>
```

5. For the mandatory `name` parameter, enter the name of the cookie.
6. For the mandatory `value` parameter, enter the string that represents the cookie
7. For the mandatory `date` parameter, enter a properly formatted date string that adheres to accepted specifications (i.e. RFC822, 850/1036, 1123, or Netscape cookie specifications). This date defines when the cookie will expire.

8. For the optional `domain` parameter, (whose default value is the host name of the server that originally sent the cookie), you can modify it to match another domain (e.g. "`.quest.com`").

   Once set, whenever a client-side user launches or installs the bundle, the cookie will be retrieved and sent to the client side.

Configuring Bundle Installation Properties

Properties that affect how a bundle is installed on the client side are typically found under the Install Data and Platform nodes in the Administration Tool. Unlike bundle installation properties that affect end-user options (which are discussed in the next section of this chapter), the options discussed here affect how administrators require the bundle to be installed. Administrator concerns include in which directory the bundle is installed, how the bundle affects the client machine's CLASSPATH and registry settings, and which of the bundle's files are the readme and license files.

Some bundle properties must be defined in order for proper installation to occur. The Administration Tool verifies that these essential properties have been defined before the bundle can be saved and uploaded to a server.

Setting Bundle Install Directories

The bundle's location on the client machine is partially determined by values you enter for the Vendor and Platform installation directories. A bundle's Vendor Directory, Platform Install Directory, and the User Queries Install Directory all affect how a bundle is installed and launched. (Please refer to Bundle Installation Directories: Creation Strategies later in this chapter for more information on using these properties together.)

Setting Vendor install directory

1. Expand the Install Data node.
2. Select the Vendor node.
3. In the right pane, in the Directory field, enter the Vendor Directory (without any drive letters) in which the bundle will be installed.

   The same directory should be set for all the same bundle versions, ensuring that all versions are installed in the same directory on the client machine.

   During installation, the Vendor Directory is appended to the User Queries Install Directory (if enabled), or the Platform Install Directory.

4. Continue to set other bundle properties, then click File > Update to commit this change.

Setting Platform-specific install directory

1. Expand and select the Platform All node.
2. In the right pane, in the Install Directory field, enter the default directory (including a drive letter for Windows clients) in which the common bundle files are installed.

   If no Platform Install Directory is specified, default directories are used (`c:\Program Files` on Windows, `/(root)` on Unix).
3. If bundle contents need to be installed in a platform-specific directory, the Platform node contains other platform child nodes (e.g. Windows/i386, Unix/sparc), each of which contains its own Install Directory field.

   If during installation, the user is queried for an install directory, the entered path replaces this value. Additionally, the Vendor Directory path is appended to this directory.
4. Continue to set other bundle properties, then click File > Update to commit this change.

Designating License and Readme Files

If your bundle contains license and readme files, you can indicate which files serve these roles. When a bundle is being installed by a client-side user, the license file you select will be displayed before the bundle installation begins. Similarly, the readme file you select will be displayed at the end of the installation. (You can also determine whether or not the user will have an opportunity to decline viewing the readme file. This is covered in Configuring End-User Bundle Installation Options, found later in this chapter).

Designating a license file in a bundle

1. Ensure that all the desired files (including the license file) have been added to the bundle.
2. Click the Install Data node.
3. In the right pane, click the License File combo box, and select your license file from the list of files that are part of the bundle.
4. Continue to set other bundle properties, then click File > Update to commit this change.

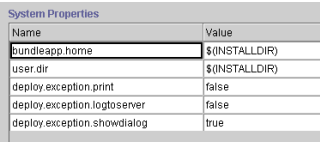
Designating a readme file in a bundle

1. Ensure that all the desired files (including the readme file) have been added to the bundle.
2. Click the Install Data node.
3. In the right pane, click the Readme File combo box and select your readme file from the list of files that are part of the bundle.
4. Continue to set other bundle properties, then click File > Update to commit this change.

Determining how Bundles Affect Client Machine Settings

If settings on the client machine (e.g. registry or CLASSPATH settings) need to be added or modified in order for the bundle application to run properly, a bundle can be configured to have these changes made upon installation.

In addition to registry and CLASSPATH values, DeployDirector System Properties can also be set, which are environment variables that are passed from the CAM to the application when it begins running. Clicking a bundle's Install Data node, then System Properties node reveals all the settings for that particular bundle version:



One property value that is typically used is `$(INSTALLDIR)`, which is the home location of the application that has been deployed (i.e. [`installdir`]/[`vendorname`]/[`bundlename`]). It is a macro that is replaced with the directory path on which the application was installed on the client side.

You can also use the `user.dir` Java system property as a DeployDirector system property. Setting this to `$(INSTALLDIR)` ensures the bundle's current directory is its install directory. If your bundle requires files at run-time, and you know their location relative to the install directory, using this setting helps maintain system focus on the correct directory. This property is also particularly useful when using the `/launch` request (please see Launching Applications, later in this chapter).

System Properties allows you a measure of control over how exceptions are handled by the CAM. The following table lists the properties that can be set using the Administration Tool. (These properties are independent of each other.).

Name	Result	Value (option/**default**)
deploy.exception.showdialog	errors sent to dialog box	false / **true**
deploy.exception.print	errors sent to the console (if present)	false / **true**
deploy.exception.logtoserver	errors written to the Client Log	false / **true**
deploy.exception.locallog	errors logged to a client-side file, located at the root of the bundle directory	<filename> / **null**
deploy.stdout.locallog	in addition to the console, output is logged to a client-side file, located at the root of the bundle directory	<filename> / **null**

Setting system properties for a bundle

1. Ensure that all files have been added to the bundle.
2. Expand the Install Data node.
3. Select the System Properties node.
4. Click Add to reveal property fields for a new system property entry.

   Any system environment variables that need to be set in order for the application to run can be defined here.
5. In the Name field, enter the name of the system property (e.g. `app.home`).
6. In the Value field, enter the system property value (e.g. `$(INSTALLDIR)`).
7. If required, continue to define more System Property entries, as well as other bundle properties, then click File > Update to commit these changes.

Setting CLASSPATH information for a bundle

1. Ensure that all files have been added to the bundle.
2. Expand the Platform All node.
3. Expand the Java node.
4. Select the Classpath node.
5. In the right pane, click Add to reveal a new CLASSPATH entry.
6. Click the Path field. It acts as a combo box, and displays all JARs and ZIP files amongst the cross-platform files that you included under the Platform All node.
7. From the list, select the JAR you would like to add to the CLASSPATH.
8. Continue to add all CLASSPATH entries that the bundle requires in order to execute properly (as an alternative, you can click Add All JARs to automatically add an exhaustive list of CLASSPATH entries).
9. If required, continue to define other bundle properties, then click File > Update to commit these changes.

Registering and creating Windows registry entries for a bundle

1. Select the Install Data node.
2. In the right pane, enable the Register Application with OS check box.

   Enabling this option places the bundle in the Windows Add/Remove Programs control panel.
3. In the left pane, select the Windows Registry node.
4. Click Add to reveal property fields for a new registry entry.
5. In the HKEY combo box, select the high level key of which this entry is a component.
6. In the SKEY field, enter the entry's subkey.
7. In the Name field, enter the name (if required).
8. In the Value field, enter the string value for entry. (Only string values are accepted.)
9. If required, continue to define more Windows registry entries, as well as other bundle properties, then click File > Update to commit these changes.

Configuring End-User Bundle Installation Options

Bundles can be configured to query client-side users during the installation of a bundle. Specifically, users can be asked where on their local file system they would like the bundle to be installed (the User Queries Install Directory), and whether they would like desktop shortcuts to be installed. Shortcuts can point to a Java file (i.e. a class defined as an Entry Point), or a support file (e.g. PDF, HTML, or text file).

Permitting users to specify an install directory

1. Expand the Install Data node, and select the User Queries node.
2. In the right pane, select the Install Directory check box, configuring the bundle to query the user for one during installation. (The bundle's Vendor Directory is appended to this directory.)

   If the Install Directory check box is disabled, the bundle's install path will consist of the Platform Install Directory and the Vendor Directory. Please refer to Configuring Bundle Installation Properties for more information on these bundle properties.

   Vendor Directory, Platform Install Directory, and the User Queries Install Directory all affect how a bundle is installed and launched. Please refer to Bundle Installation Directories: Creation Strategies for more information on using these properties together.
3. Continue to set other properties, then click File > Update to commit.

Setting bundle desktop shortcut properties

1. Ensure that all files have been added to the bundle.
2. Ensure that at least one of these files has been designated as an entry point for the bundle.

   The number of entry points you set up depends entirely on how many of your bundle's files are meant to be executable files on the client side. Setting entry points is covered in the section entitled Defining Entry Points in Configuring Bundle Runtime Properties.
3. Expand the Install Data node.
4. Select the Shortcuts node.
5. In the right pane, click Add to display the property fields for the new shortcut.
6. In the Type combo box, indicate whether the shortcut being created will point to a Java class, File, Update script, or Uninstall script.
7. In the Name field, enter the name with which the shortcut will be referred on the client machine.

8. In the Link combo box, select the entry point that is associated with this shortcut.

   The information available in the Link list depends on the shortcut Type you chose in a previous step. If you chose a Java shortcut, the Link list displays only defined entry points. If you chose a File shortcut, you need to indicate which bundle file is associated with the shortcut. If Update or Uninstall shortcuts are used, a Link definition is not necessary, since the client-side functionality for these actions are automatically handled by DeployDirector.

9. From the Desktop combo box, select whether or not the shortcut is meant to appear on the client machine desktop in addition to its Start Menu (for Windows clients).

10. In the Win Icon combo box, select the bundle file that is the shortcut image (which is typically an `.ico` or `.dll` file).

11. If a `.dll` file was selected from the Win Icon combo box in the previous step, enter the index number of the desired icon in the Win Icon Index field.

    Entering proper values for these fields and combo boxes defines a shortcut profile, where a Start Menu shortcut (and if enabled, a desktop shortcut), using the defined icon image, is created on the client machine.

    The choice over the creation of a desktop shortcut (assuming a value of True was selected from the Desktop combo box) can be given to the user.

12. Select the User Queries node.

13. In the right pane, enable the Desktop Shortcuts check box if you want the user to choose whether or not one is created. Disabling the check box will leave this choice to you. (The choice made in the Desktop combo box, under the Shortcuts node earlier in this procedure determines this.)

14. Continue to set more shortcuts, as well as other bundle properties, then click
    File > Update to commit these changes.

Bundle Installation Directories: Creation Strategies

In earlier sections (Configuring Bundle Installation Properties, and Configuring End-User Bundle Installation Options), you learned how to set the Vendor Directory, Platform Install Directory, and User Queries Install Directory. Each of these directory values can contribute, through appending or replacement, to a bundle's final installation path on a client machine. While the presence of three directory-related bundle properties may necessitate extra thought in how they are defined, the result is a large amount of flexibility with how bundles can be installed on different types of machines. The following table outlines the three bundle directory properties

	Vendor Directory	Platform Install Directory	User Queries Install Directory
Where it is set	Install Data node > Vendor child node > Directory property text field.	Platform node > Install Directory property text field (can also be set in other Platform child nodes).	Install Data node > User Queries child node > Install Directory check box.
Function	Sets the vendor (i.e. bundle creator) directory.	Sets an platform-specific install paths.	When enabled, allows the end user to enter an installation path.
Additional Info	Mandatory setting. Same directory value should be set for all versions of the same bundle.	For Windows clients, must contain a drive letter. If not set, default directories include c:\Program Files on Windows, and /(root) on Unix.	If enabled, user-inputted directory path overrides Platform Install Directory information.
How it affects the bundle's installation path	*Always* appended to the Platform Install Directory, or User Queries Install Directory.	Acts as first half of installation directory path, but replaced if user inputs directory.	If enabled, user-inputted directory replaces Platform Install Directory. Launch command functionality cannot be used.

How these bundle properties are configured depends on your organization's deployment strategy. Whether you require heavily or lightly controlled installations, bundles can be configured to match that need.

Enforcing Strict Bundle Installation Paths

Keeping complete control over numerous versions of a bundle to numerous clients has obvious benefits. You can configure bundles so that they are installed in a controlled manner; one that allows system administrators to know exactly where bundles and bundle versions can be found on client machines.

When you configure bundle properties that affect installation directories, hard code install directories, and disable user-inputted paths. Consider this example:

Bundle Property	Setting
Vendor Directory	Vendor
Platform Install Directory	D:/InstallHere
User Queries Install Directory	disabled
Bundle install directory on client machine	D:/InstallHere/Vendor/BundleName

When installing a bundle with these settings, the user will be shown the installation path, but they will not be able to change it.

Allowing User-Defined Installation Paths

If your organization does not need to keep a very close eye on where client-side users are installing applications, then you can give users complete control over where bundles are installed. Consider this example:

Bundle Property	Setting
Vendor Directory	Vendor
Platform Install Directory	none
User Queries Install Directory	enabled
Bundle install directory on Windows client machine	C:/Program Files/Vendor/BundleName or C:/User/Defined/Directory/Vendor/BundleName

During installation of this bundle, the install directory is presented to the client-side user, who has an opportunity to change it. If they continue without changing it, the installation path consists of the default Platform Install Directory and Vendor Directory. However, if the user modifies the path, their inputted directory path replaces the default directory, and the vendor directory is then appended to it.

Configuring Installation Directories for Use with the Launch Command

The `/launch` request, discussed later in this chapter, gives end users easy access to applications that have been installed on their Windows-based client machines (e.g. a one-click application launch from an internal Web page). If a requested bundle is not present on the client machine, it is first installed, then run.

Given this launch/install behavior, DeployDirector must know exactly where the bundle can be found on client machines if it is to determine whether or not it exists. Thus, the only way to ensure that bundles work with the `/launch` command is to hard-code installation directory paths by disabling the User Queries Install Directory.

As an example, consider a bundle that does not yet exist on a client machine, and is called with the `/launch` command: it will first be installed. However, if an end user is given an opportunity to request the bundle with the `/install` command, and has the option to define where it is installed, problems may ensue if the `/launch` command is used afterwards. Consider a bundle whose install directory properties have been configured to query the end user for an install directory:

Bundle Property	Setting
Vendor Directory	Vendor
Platform Install Directory	none (defaults to C:/Program Files)
User Queries Install Directory	enabled (user enters C:\User\Defined\Directory)
Bundle install directory on Windows client machine	C:/User/Defined/Directory/Vendor/BundleName

If this bundle is called with the `/launch` command, it will first be installed. Even though the bundle has been configured to query the user for an install directory, the `/launch` command automatically skips that step.

However, if the bundle is called with the `/install` command, during installation, the user will have an opportunity to enter a directory. This inputted path (in this case, `C:\User\Defined\Directory`), will replace the Platform Install Directory (in this case, the default `C:\Program Files`).

When the user launches this application in the future, using information found in the bundle's settings, DeployDirector will look in `C:/Program Files/Vendor/` for the bundle. Since the user installed it elsewhere, the application will not be found, and will be installed again in the searched directory.

This problem can be avoided if either the user is only given access to applications with the `/launch` command, or bundles are configured so that the User Queries Install Directory property is disabled.

Extending Installation Options with Custom Classes

During the installation, updating, and uninstalling of applications with DeployDirector, installation events are generated to pass information about the state of these processes. The `InstallEvent` class is used to detect install events and pass information to a complementary listener class. These two classes can be used to customize the bundle's installation process.

Using the DeployDirector API, an administrator can create a registered `InstallListener` class which will listen for `InstallEvent` objects so that the customized code can be executed in response to these events. For example, depending on the needs of your organization, this feature can be used by the network administrator to notify the end user about the status of the install process, display a custom splash screen on the client side just before beginning an installation, display a custom dialog box, or move files around after installation.

The custom class or JAR which you create has to implement the `InstallListener` class. The created custom class or JAR must be added to the DDCAM bundle and to the CAM CLASSPATH, so that it can be used when the CAM is started. This first step ensures that the custom class or JAR will be available before the bundle is downloaded to the client. Your custom class must then be specified in the bundle as a recipient of installation events.

The `InstallListener` interface can be found in the `com.sitraka.deploy` package of the DDSDK. For further details on using `InstallListener` and `InstallEvent`, please refer to the API Documentation and the DeployDirector SDK Programmer's Guide, both of which are provided with the DDSDK bundle installation.

Here is an overview of the process:

1. Create a custom class or JAR that implements the `InstallListener` class.
2. Add the class or JAR you created to the DDCAM bundle and its CLASSPATH.
3. Specify the class you created to the application that will be deployed to client desktops.

Adding the customized class or JAR to the DDCAM bundle and its CLASSPATH

1. In the Administration Tool, click the Bundles tab.
2. Expand the DDCAM node and click the current version.
3. Click the Copy Server Version button.

   This will create a new bundle version based on the most recent version in the vault.
4. Enter the version name in the Copy Server Version dialog box.
5. Click the new DDCAM version you created in the preceding step. Add your comments about this version of the DDCAM in the Description field in the right pane.
6. Expand this DDCAM version node.
7. Select the Platform All node.
8. Click the Add Files button.
9. In the Select Files dialog box, ensure that you set the correct Source Directory to the class you have created and click Open. If you have created a JAR, add it to the DDCAM bundle.
10. In the Platform All > Java > Classpath node, add the custom class or JAR to the DDCAM's CLASSPATH.
11. Commit the changes to the server by clicking File > Update Server.

The next section outlines how to specify the custom class, which implements the `InstallListener` class, to the bundle that will be deployed across the network to the client desktops. In both cases, please do not commit the changes to the server until you have completed configuring the bundle.

Specifying the custom class to your bundle

1. Expand the version node of your bundle.
2. Click the Install Data node.
3. In the Install Event Class field, enter the full package name of your custom class that implements the `InstallListener` class (you do not need to specify the extension `.class`).

   Ensure that you have added your class or JAR to the DDCAM and its CLASSPATH, as described in the "Adding the customized class or JAR to the DDCAM bundle and CLASSPATH" section.
4. Continue to set other bundle properties, then click the Update Server button or File/Update to commit the changes to the server.

Chapter 6
Configuring Bundle Runtime Properties

Whereas in previous chapters, you learned how to define bundle content as well as installation properties, this chapter focuses on how bundles act after they are installed. Specifically, certain properties affect how a bundle or application behaves when it is started up by client-side users. Runtime behavior begins with a bundle's defined entry point (the file that runs the application), working through JRE and VM issues, as well as user authentication and authorization. Security and exception handling and output are ongoing runtime issues that can also be configured.

Defining Entry Points

When you add files to a bundle, it is important to flag certain files whose roles need to be known in order for the bundle to be deployed and executed successfully. In addition to designating a particular bundle file as the readme, and another as the license information, you can also define which files are the entry points to the application.

Entry points indicate which class is executed by the CAM when the client-side user wishes to run the application in the deployed bundle. Generally, setting up an entry point should also be accompanied by the definition of a shortcut for that same file, so after installation, the end-user can easily run the application. (For more information about setting shortcuts, please see Configuring End-User Bundle Installation Options in Chapter 5, Configuring Bundle Installation Properties.)

Designating Bundle Files as Entry Points

1. Ensure that all files have been added to the bundle.
2. Expand the Platform All node.
3. Expand the Java node.

4. Select the Entry Points node.
5. In the right pane, click Add to display the property fields for a new entry point.
6. In the Name field, enter a name for this entry point.

   Make a note of the name given to the entry point, as you will need to select it if defining desktop shortcuts. (Please see Configuring End-User Bundle Installation Options in Chapter 5 for more information.)
7. In the Class field, enter the package and name of the class you want to designate as an entry point, ensuring that you *exclude* the `.class` extension (e.g. `com.company.package.Main`). This class should be among those already included in the bundle.

   It is important to ensure the package and class match that of the file in your bundle. Otherwise, the CAM will not be able to locate it and generate an executable file.
8. In the Arguments field, enter any execution arguments for the entry point if required.

   Note: If you update the arguments for a newly created version of the bundle, the end user must allow the application to be updated, then close and re-launch the application. Otherwise, the application will be launched with the previous version's arguments.
9. If required, continue to define more entry points, as well as other bundle properties, then click File > Update Server to commit these changes.

Bundle JRE Requirements

As your organization develops applications, it is likely that different apps will have different JRE requirements. You can define a bundle so that its JRE requirements (i.e. a particular version of a specific vendor's JRE) are known to DeployDirector, which the latter uses during deployment.

Without knowing which JRE vendors and versions are available on all client machines, DeployDirector eliminates second-guessing by sending out the required JRE when deploying a bundle. (JREs required by all bundles are stored on the server.) The enforced sending out of the required JRE ensures that the deployed application will run. However, to avoid redundancy, DeployDirector will ensure bundles that have the same JRE requirements, and are located in the same Vendor install directory, have not been deployed to the target client machine first. If this is the case, the JRE is not sent.

Checking for JREs on the Client Side

Even though DeployDirector can deploy required JREs from its vault to client machines, and the CAM can always search for these vault-based JREs when installing a bundle, it is possible that an appropriate JRE already exists (e.g. one that came with the client machine's OS).

When configuring a bundle, you can enable a Search for Installed JREs operation, which attempts to find an installed JRE that is used by the bundle. If an appropriate match is found, a duplicate JRE is not deployed from the vault, reducing download time.

Deciding whether you want the CAM to search for an existing JRE that was not deployed by DeployDirector depends on how important reduced download times as well as smaller installation footprint (particularly where multiple JREs are installed in different Vendor directories) are in relation to increased quality control.

While using the Search For Installed JREs option can save time, disabling it ensures tighter control over client machine profiles. For example, if a JRE not associated with DeployDirector is used but is then subsequently removed, the bundle will no longer work. JREs installed by DeployDirector cannot be removed using legitimate OS methods (e.g. the Add/Remove Programs Control Panel), ensuring that required JREs will always be available to bundles, and will not incidentally be removed by other application installation/removal tools.

Setting the JRE properties for a bundle (including VM parameters)

1. Ensure the required JRE is present on either the client machine or in the vault.
2. Determine the minimum or specific JRE requirements for the application in your bundle.
3. Select the Install Data node.
4. In VM Parameters field in the right pane, enter any parameters that the JRE requires to successfully run the bundle. The information entered in this field is appended to the JRE execution command.

   You may wish to set the `-noverify` parameter here if your bundle will be used with a 1.1 VM. Please refer to Class Verification and Using the -noverify VM Parameter discussed later in this chapter for more information.
5. Expand the Platform All node.
6. Select the Java node.
7. If the required JRE comes from a specific vendor (i.e. IBM or Sun), select it from the Vendor combo box.

8. In the Version field, enter the JRE version required by the application.
9. In the Version combo box, select the condition that matches the exclusiveness of the JRE version definition.
10. Select the Share VM check box if this bundle is meant to share a VM with any other bundle that also has this feature enabled, and is using the same JRE. Disable the check box if the application is meant to always run using its own JRE.

    You should be aware of how DeployDirector handles shared VMs before setting this property, as well as conflicts with PLAF for third party libraries. Please refer to the next section, Sharing VMs Between Multiple Applications, for more information.
11. Enable or disable the Search for Installed JREs check box depending on whether the CAM should first search on the client machine for an appropriate JRE that was not installed by DeployDirector.
12. Continue to set other bundle properties, then click File > Update Server to commit this change.

Sharing VMs Between Multiple Applications

DeployDirector helps save system resources on the client side by allowing multiple applications to use a shared VM, provided that the VM versions required by the applications are compatible and a shared CAM is used. The Share VM check box can be found under the Bundle_Name > Version > Platform(All) > Java node. If you select this check box while configuring the properties of the bundle, then this particular bundle will share a VM with any other deployed, running application with an enabled Share VM property.

This feature is not intended to improve the start up time, but it is meant to save system resources on the client machine and to allow updating and restarting the application without restarting the VM. When your application is run on the client side, its own VM originally starts, but the CAM searches for other active identical VMs. If one is found, the first VM is then shut down and the responsibility of running the application is transferred to the other VM.

If you are creating bundles that are meant to share the same VM, they must be configured to be installed in the same vendor directory on client machines. (For information on setting the vendor directory, please see Setting Bundle Install Directories, in Chapter 5, Configuring Bundle Installation Properties.)

Sharing VMs: the Effect on the CAM's Class Loader

When running a bundle, the setting of the Share VM property affects which class loader (i.e. custom class loader or system class loader) is used by DeployDirector for the application classes. When Share VM is enabled, DeployDirector uses its custom class loader which allows a deployed application to shut down completely and removes any file system locks on the class and JAR files from which the application was loaded.

When this custom class loader is used, only the CAM classes are available to the system class loader. Consequently, if any class in the application requests the system class loader to load other application classes or resources, for example by calling `java.lang.ClassLoader.getSystemClassloader()`, a `ClassNotFoundException` or similar will be generated.

This situation is not common, but occasionally occurs in third-party libraries which unnecessarily use the system class loader. It also occurs when using custom PLAF (Pluggable Look and Feel) classes due to an unfortunate oversight in the current implementation of Swing. In these situations, disabling Share VM (thus using the system class loader) will work around the problem.

The Share VM Property and the System Class Loader

If the client application implements custom PLAF (Pluggable Look and Feel) classes or any classes/third party libraries that directly request the system class loader, ensure that you leave the Share VM box unchecked (default setting) when configuring the properties for the bundle.

Class Verification and Using the -noverify VM Parameter

Class verification ensures that all classes referenced, and interfaces implemented by a loaded class have also been loaded.

By default, JDK 1.1 VMs allow you to use lazy verification. In this case, if an application's code references a class that is not accessed during the execution of the program, this referenced class is not searched for. However, using the `-verify` flag forces recursive verification of all referenced classes.

For JDK 1.2 (or later) VMs, lazy verification does not exist. All classes referenced anywhere in the application code must be available to be loaded and verified when the referring class is loaded.

Despite the availability of lazy verification in 1.1 VMs, aggressive verification is triggered when bundles are accessed with DeployDirector due to a JDK 1.1 bug. You need to configure a bundle to use the `-noverify` VM parameter if you wish to use, or are relying on lazy verification with applications using the 1.1 VM.

Using the `-noverify` parameter will also dramatically improve the startup time of JDK 1.1 apps that use Swing. It is recommended that this parameter is set for all applications that use the JDK 1.1 VMs, unless aggressive verification is needed. VM parameters for a bundle can be set under the Bundle_Name > Version > Install Data node. (Please refer to the procedure entitled Setting the JRE properties for a bundle (including VM parameters), for information on how to set this property for a bundle.)

Client-Side Exception Handling and Output

By default, bundles are configured so that exceptions are sent to the console window (if present), reported to the end user with a dialog box, and written to the server's Client Log. Additionally, standard output is sent to the console window.

Bundles can be configured so that standard errors are sent to any number of these destinations, or error reporting can be turned off. Standard errors and output can also be logged to separate client-side files. All of these properties are defined in a bundle's System Property node (please see Determining how Bundles Affect Client Machine Settings in Chapter 5, Configuring Bundle Installation Properties).

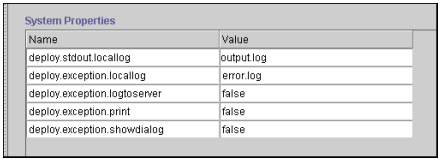
Configuring Standard Exception and Output Destinations

Error reporting destinations can be set independently of each other. When configuring a bundle, if you want to disable error reporting in the console, dialog box, or Client Log, its property needs to be given a `false` value. In order to write standard errors or output to a client-side file, you need to provide a file name. All of these are properties and values are possible entries for the System Property node:

Name	Result	Value (option / **default**)
deploy.exception.showdialog	errors sent to dialog box	false / **true**
deploy.exception.print	errors sent to the console (if present)	false / **true**
deploy.exception.logtoserver	errors written to the server's Client Log	false / **true**
deploy.exception.locallog	errors logged to a client-side file, located at the root of the bundle directory	<filename> / **null**

Name	Result	Value (option / **default**)
deploy.stdout.locallog	in addition to the console, output is logged to a client-side file, located at the root of the bundle directory	<filename> / **null**

In the following example, all standard errors will be reported in a client-side file (`error.log`) only. In addition to the console, standard output will also be written to a client-side file (`output.log`):


System Properties

Name	Value
deploy.stdout.locallog	output.log
deploy.exception.locallog	error.log
deploy.exception.logtoserver	false
deploy.exception.print	false
deploy.exception.showdialog	false

Note: Bundles that have been configured to report standard errors and/or output in client-side files, must not share the same VM. If configuring a bundle to write to files, disable its Share VM property (found in the Platform (All) > Java node for that bundle version).

End-User Authentication and Authorization

In this chapter, authentication and authorization are discussed in the context of a bundle's runtime properties. Specifically, the following sections center around the configuration of properties that affect how authentication and authorization occur when an end-user attempts to run it on the client side.

In addition to runtime settings, there are other important concepts related to authentication and authorization, and how the Administration Tool can be used to manage end-user lists, and create associations between bundles and users. Please refer to An Overview of User Authentication and Authorization in Chapter 9 for more information.

The Authentication and Authorization Process

While different authentication and authorization classes exist (whether they are the default, or your own customized classes), the process of allowing a user to download and install a bundle follows the same general pattern:

1. The authentication and authorization process is initiated when a client-side request for a bundle or bundle update is sent to a server.

   The bundle's Connection property settings determine whether this request is initiated by the user, or sent automatically by the bundle.

2. The transfer of authentication information begins on the client side. It is either entered by the end user and passed to the CAM, or the CAM receives the user request and retrieves cached user authentication information.

   All default DeployDirector authentication and authorization classes use caching. If authentication information is required by the user, they are only required to provide it once (i.e. the first time they request a bundle).

3. The user authentication information is transferred to the server side (SAM), where its validity is verified.

4. After the user has been authenticated, the SAM ensures they are authorized to access the bundle or bundle version they are requesting.

5. If the user is authorized to access the bundle or bundle version, the SAM initiates the deployment of that bundle.

At any point during this process, if the authentication or authorization information does not match with user information stored on the client or server side, the CAM informs the user that they are not able to download and install the bundle they are requesting.

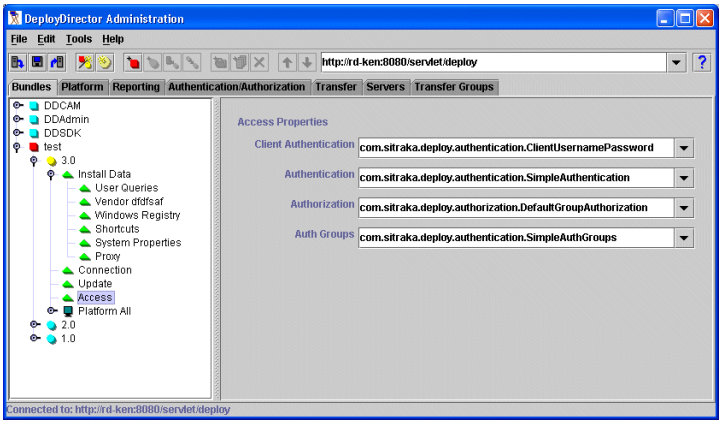
Obviously, the finer details of this authentication and authorization process depend on the classes you use to carry it out, and the properties you set for those classes in the Administration Tool.

Setting Authentication Properties

In DeployDirector's Administration Tool, you can easily set authentication properties for bundles. This is performed by choosing specific module (and corresponding editor classes) that are 'plugged in,' and setting any required properties in editors based in the Administration Tool.

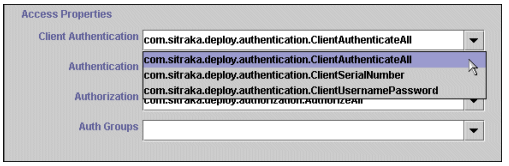
When working with a new bundle or bundle version, selecting its Access property node displays combo boxes for both authentication modules (client-side authentication, and server-side authentication), as well as those for authorization.

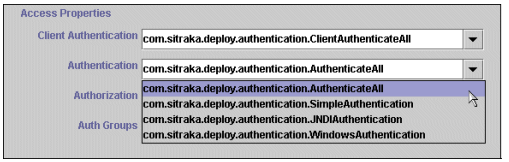
In the module's property field in the right pane, you can enter the full package name of the desired class, or select a class from the drop-down list:

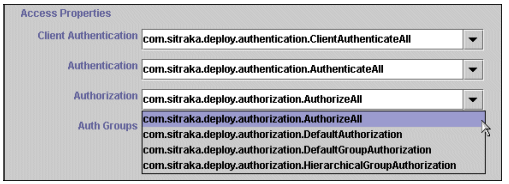


Selecting classes in these Access property fields results in changes to the bundle's `version.xml` file. Corresponding to the properties structure in the Administration Tool, authentication and authorization properties are encompassed by the file's `ACCESS` tag, which contains sub-tags that are associated with the classes you choose.

For example, making these class selections for the three access modules in the Administration Tool would result in the following `version.xml` entry:








```
<ACCESS
ALLOWCACHE="true"
CLIENTAUTHENTICATION="com.sitraka.deploy.authentication.
    ClientAuthenticateAll"
AUTHENTICATION="com.sitraka.deploy.authentication.
    AuthenticateAll"
AUTHORIZATION="com.sitraka.deploy.authorization.
    AuthorizeAll"/>
```

Allowing automatic authentication of all users

1. Select the bundle version's Access node to reveal all of its access properties in the right pane.
2. In the Client Authentication drop-down list, select the `ClientAuthenticateAll` class (whose full package is `com.sitraka.deploy.authentication.ClientAuthenticateAll`).
3. In the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `AuthenticateAll` class.
4. Continue to configure your bundle, then commit these changes to the server.

Authenticating users by matching server data with client names and passwords

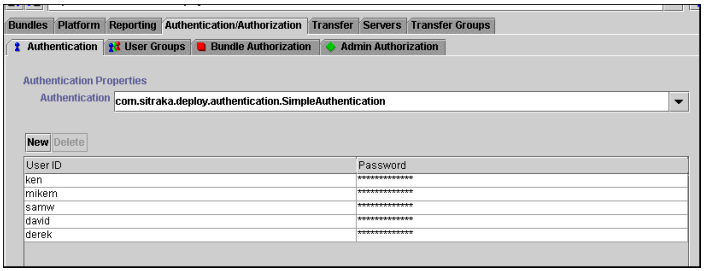
1. Select the bundle version's Access node to reveal all of its access properties in the right pane.
2. From Client Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `ClientUsernamePassword` class if user authentication information should consist of a user name and password.
3. From the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `SimpleAuthentication` class.

   After selecting this class, you may want to review the contents of your server-side data file with the Simple Authentication editor.
4. Click the Authorization/Authentication tab, then click the Authentication child tab.
5. From the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `SimpleAuthentication` class.

   Selecting this class displays a list of valid user authentication information.

Important: The information displayed in the list is locally cached, thus is as current as the last time you refreshed it. It is important to refresh lists when using this editor.

Please refer to An Emphasis On Server Updating and Refreshing in Chapter 9 for more information.



6. Click the Refresh button (or, File > Refresh) to ensure the list currently reflects the contents of the server data file.
7. If required, review and edit the User ID and Password fields to match any new changes in your organization's pool of users. If you edit any fields, press the Enter key to commit it.

   You can learn more about managing lists of authenticated users in End-User and Administrator Authentication Lists in Chapter 9.
8. Continue to configure your bundle, then commit this change to the server.

Authenticating users by matching server data with client serial numbers

1. Select the bundle version's Access node to reveal all of its access properties in the right pane.
2. From Client Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `ClientSerialNumber` class if you want a user's identification key (typically just a user ID), but do not require a password as their authentication information.
3. From the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `SimpleAuthentication` class.

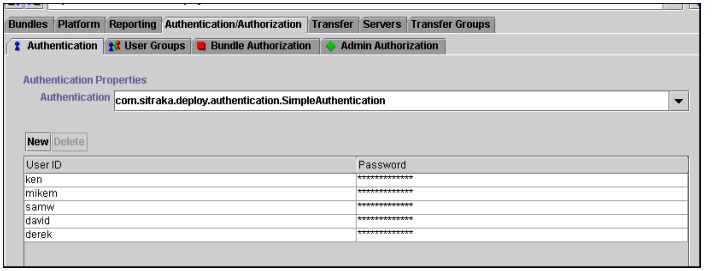
   After selecting this class, you may want to review the contents of your server-side data file with the Simple Authentication editor.
4. Click the Authorization/Authentication tab, then click the Authentication child tab.

Important: The information displayed in the list is locally cached, thus is as current as the last time you refreshed it. It is important to refresh lists when using this editor.

Please refer to An Emphasis On Server Updating and Refreshing in Chapter 9 for more information.

5. From the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `SimpleAuthentication` class.

   Selecting this class displays a list of valid user authentication information.



6. Click the Refresh button (or, File > Refresh) to ensure the list currently reflects the contents of the server data file.
7. If required, review and edit the User ID and Password fields to match any new changes in your organization's pool of users. If you edit any fields, press the Enter key to commit it.

   You can learn more about managing lists of authenticated users in End-User and Administrator Authentication Lists in Chapter 9.
8. Continue to configure your bundle, then commit this change to the server.

Authenticating users against a directory such as NIS

Important: Using NIS authentication in this context requires the use of two specialized JARs. Please refer to Server-Side Authentication Module and Editor Classes on page 156 for important information.

1. Ensure the CLASSPATH includes `NIS.jar` and `ProviderUtil.jar`, or if you are using the standalone server, ensure these JARs have been placed in the `deploydirector/lib` directory.

2. In the Administration Tool, select the bundle version's Access node to reveal all of its access properties in the right pane.

3. From the Client Authentication drop-down list, select the `ClientUsernamePassword` class (whose full package is `com.sitraka.deploy.authentication.ClientUsernamePassword`).

   Since authentication is NIS-based, there is no need for end-users to provide information themselves.

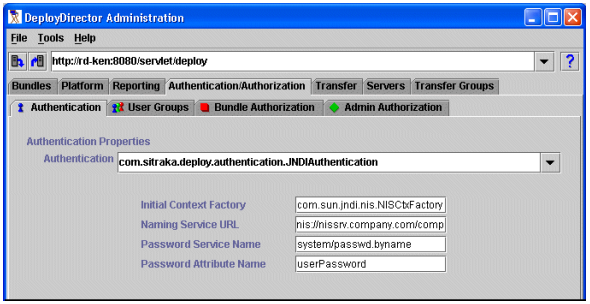
4. From the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `JNDIAuthentication` class.

   Selecting this Access Properties class means that the JNDI Authentication editor needs to be reviewed.

5. Click the Authorization/Authentication tab, then click the Authentication child tab.

6. From the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `JNDIAuthentication` class.

   Selecting this editor reveals the current status of the configuration file for JNDI authentication. This server-side file points to the NIS naming service, the JNDI-based authentication mechanism in DeployDirector.



7. In the Initial Context Factory field, enter the name of the class (with full package) that is used to select the NIS provider.

8. In the Naming Service URL text field, enter the NIS server URL. Valid formats include:

   ```
   nis://<hostname>/<domainname>
   nis:///<domainname>
   nis:/<domainname>
   nis:<domainname>
   ```

   The host name or IP address of the server `<hostname>` that is offering the NIS service to a domain `<domainname>` must be defined, otherwise the process may fail.

9. In the Password Service Name field, you may enter the name for the map list that contains password information, in which the user name is the key.

10. In the Password Attribute Name field, enter the user password for this established domain.

11. Continue to configure your bundle, then commit this change to the server.

Authenticating Windows users based on their login information

1. Select the bundle version's Access node to reveal all of its access properties in the right pane.

2. From the Client Authentication drop-down list, select the `ClientUsernamePassword` class (whose full package is `com.sitraka.deploy.authentication.ClientUsernamePassword`).

   Since authentication is NIS-based, there is no need for end-users to provide information themselves.

3. From the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `WindowsAuthentication` class.

   Selecting this Access Properties class means that the Windows Authentication editor needs to be reviewed.

4. Click the Authorization/Authentication tab, then click the Authentication child tab.

5. From the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`) `WindowsAuthentication` class.

   Selecting this editor reveals the current status of the configuration file for Windows authentication. This server-side file points to a Windows

network domain, which acts as an authentication mechanism in DeployDirector.

6. Verify or enter the name or value of the Windows domain, whose registry of valid user login information is referenced when a user is being authenticated (it is this registry to which a user's provided name and password will be compared).
7. Continue to configure your bundle, then commit this change to the server.

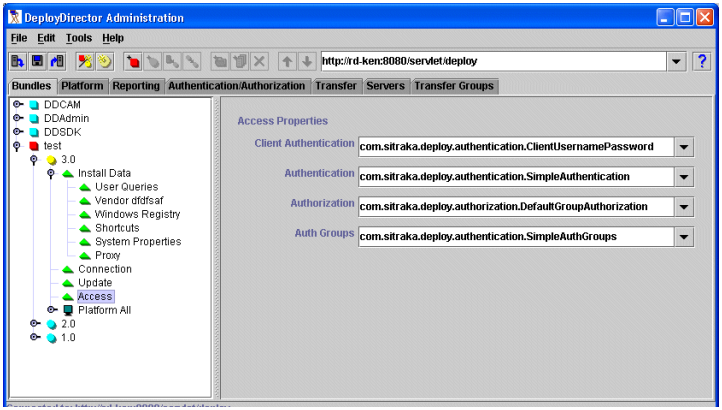
Setting Authorization Properties

As with authentication properties, you can set authorization properties in DeployDirector's Administration Tool by selecting the Access property node for a bundle, where you can choose specific modules and corresponding editor classes.

Important: When a bundle is configured to use certain default authorization modules, the formatting used for its version name (e.g. 1.2.0) must conform to certain constraints. Please refer to Authorization Behavior and Allowable Bundle Version Names in Chapter 9, for more information.

When working with a new bundle or bundle version, selecting its Access property node displays combo boxes for both authorization modules, as well as the previously discussed authentication modules. Available authorization classes include global and unconditional authorization, individual authorization, group authorization, and hierarchical group authorization. For an overview of these authorization modules and classes please see Authorization Module and Editor Classes on page 157, and Group Authorization Module and Editor Classes on page 158.

In the module's property field in the right pane, you can enter the full package name of a custom class you have created, select a DeployDirector class from the drop-down list:



Authorizing all authenticated users

1. Select the bundle version's Access node to reveal all of its access properties in the right pane.
2. Ensure you have selected the appropriate Client Authentication and Authentication classes from their respective drop-down lists.
3. From the Authorization drop-down list, select the (`com.sitraka.deploy.authorization.`) `AuthorizeAll` class.
4. Continue to configure your bundle, then commit this change to the server.

Authorizing authenticated users based on the contents of a server-side data file:

1. Select the bundle version's Access node to reveal all of its access properties in the right pane.
2. Ensure you have selected the appropriate Client Authentication and Authentication classes from their respective drop-down lists.
3. From the Authorization drop-down list, select the (`com.sitraka.deploy.authorization.`) `DefaultAuthorization` class.

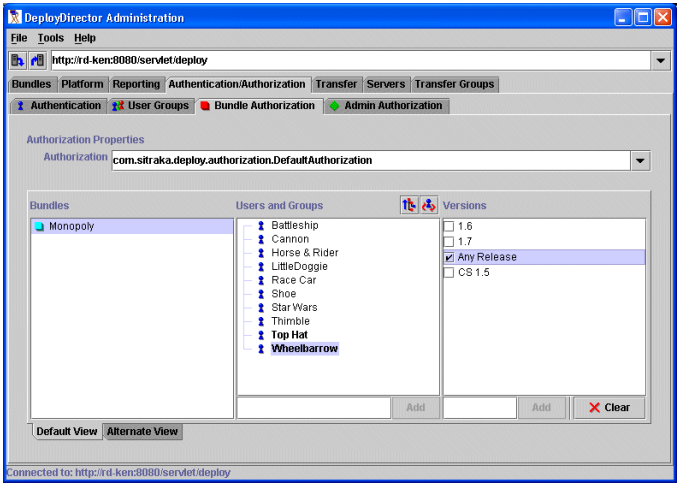
   After selecting this class, you may want to review the contents of the server-side data file in the Default Authorization editor.
4. Click the Authorization/Authentication tab, then click the Bundle Authorization child tab.

5. From the Authorization drop-down list, select the (`com.sitraka.deploy.authorization.`) `DefaultAuthorization` class.

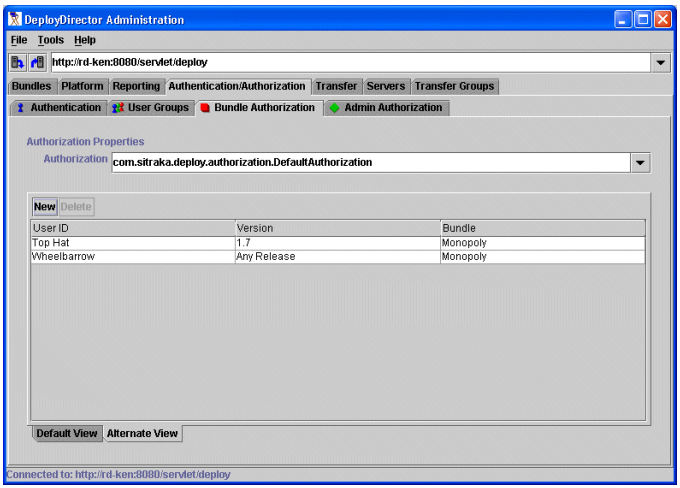
   Selecting this class shows authorization associations based on the server-side authorization data file, which is referenced by DeployDirector during authorization. This view shows associations between relevant bundles (i.e. those that use the `DefaultAuthorization` class), users and groups, and bundle versions.

Important: The information displayed in the list is locally cached, thus is as current as the last time you refreshed it. It is important to refresh lists when using this editor.

Please refer to An Emphasis On Server Updating and Refreshing in Chapter 9 for more information.



The Alternate View displays the contents of the authorization data file.



6. Click the Refresh button (or, File > Refresh) to ensure the list currently reflects the contents of the server data file.

7. Review and if required, edit the users or groups that are authorized to use the bundle version. Whenever you edit any fields, press the Enter key to commit it.

   You can learn more about authorizing users or groups to particular bundles in End-User and Administrator Authentication Lists in Chapter 9.

8. Continue to configure your bundle, then commit these changes to the server.

Authorizing groups of users based on a server-side data file

1. Select the bundle version's Access node to reveal all of its access properties in the right pane.

2. Ensure you have selected the appropriate Client Authentication and Authentication classes from their respective drop-down lists.

3. From the Authorization drop-down list, select the (`com.sitraka.deploy.authorization.`) `DefaultGroupAuthorization` class.

   The selection of a group authorization class in this field (in this case, `DefaultGroupAuthorization`) results in the enabling of the Auth Groups drop-down list is enabled.

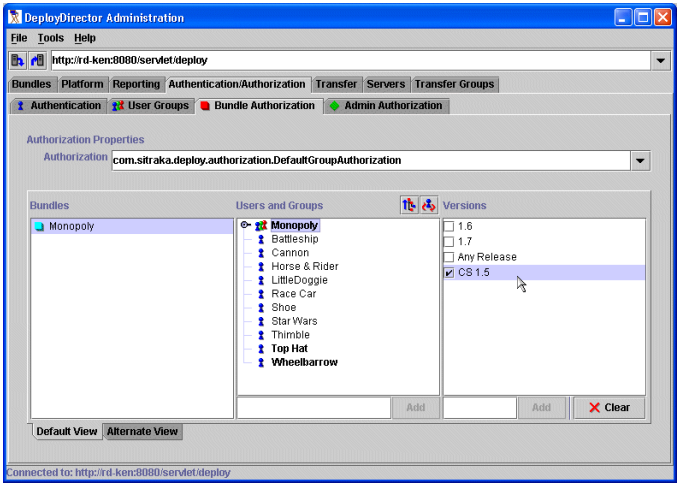
4. From the Auth Groups drop-down list, select the `SimpleAuthGroups` class.

   After selecting these classes, you may want to review the contents of your server-side data file with the Default Authorization editor.

5. Click the Authorization/Authentication tab, then click the Bundle Authorization child tab.

6. From the Authorization drop-down list, select the (`com.sitraka.deploy.authorization.`) `DefaultGroupAuthorization` class.

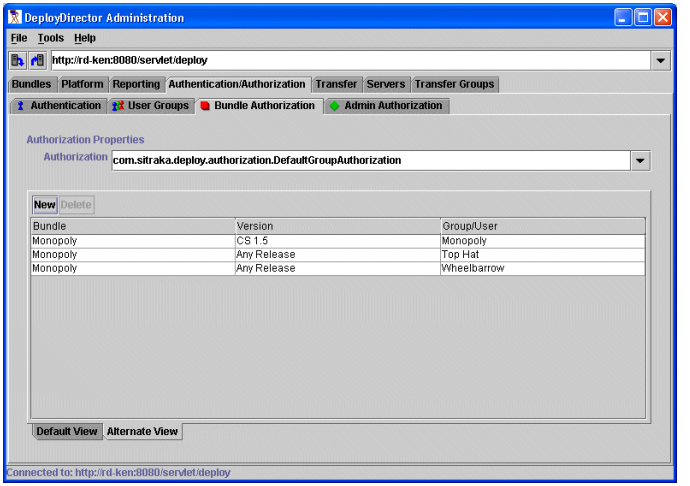
   Selecting this class shows authorization associations based on the server-side group authorization data file, which is referenced by DeployDirector during group authorization. This view shows associations between relevant bundles (i.e. those that use the `DefaultAuthorization` class), groups and users, and bundle versions.



Important: The information displayed in the list is locally cached, thus is as current as the last time you refreshed it. It is important to refresh lists when using this editor.

Please refer to An Emphasis On Server Updating and Refreshing in Chapter 9 for more information.

The Alternate View displays the contents of the group authorization data file.



7. Click the Refresh button (or, File > Refresh) to ensure the list currently reflects the contents of the server data file.

8. Review and if required, edit the groups or users that are authorized to use the bundle version. If you edit any fields, press the Enter key to commit it.

   You can learn more about authorizing users or groups to particular bundles in End-User and Administrator Authentication Lists in Chapter 9.

9. Continue to configure your bundle, then commit these changes to the server.

Authorizing groups of users against an NIS directory

Important: Using NIS authentication in this context requires the use of two specialized JARs. Please refer to Group Authorization Module and Editor Classes on page 158 for important information.

1. Ensure the CLASSPATH includes `NIS.jar` and `ProviderUtil.jar`, or if you are using the standalone server, ensure these JARs have been placed in the `deploydirector/lib` directory.

2. In the Administration Tool, select the bundle version's Access node to reveal all of its access properties in the right pane.

3. Ensure you have selected the appropriate Client Authentication and Authentication classes from their respective drop-down lists.

4. From the Authorization drop-down list, select either the (`com.sitraka.deploy.authorization.`) `DefaultGroupAuthorization` or the `HierarchicalGroupAuthorization` class.

   The selection of either group authorization class in this field (whichever you have determined to be appropriate) results in the enabling of the Auth Groups drop-down list is enabled.

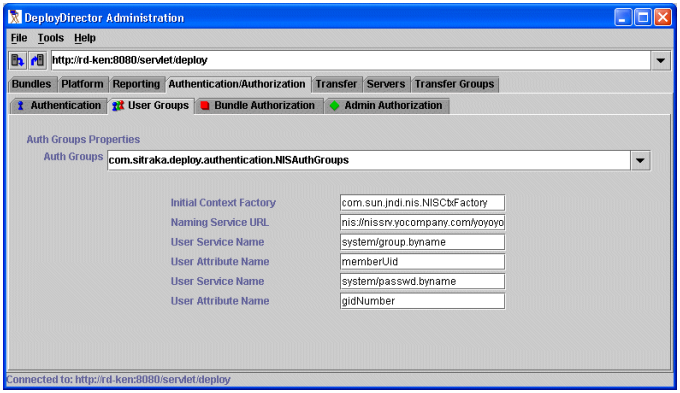
5. From the Auth Groups drop-down list, select the (`com.sitraka.deploy.authentication.`) `NISAuthGroups` class.

   After choosing this NIS-related class, you may want to review the NIS directory settings you have made.

6. Click the Authorization/Authentication tab, then click the User Groups child tab.

7. From the Auth Groups drop-down list, select the (`com.sitraka.deploy.authentication.`) `NISAuthGroups` class.

   Selecting this editor reveals the current status of the configuration file for NIS authentication. This server-side file points to the NIS naming service, and the JNDI-based authentication mechanism in DeployDirector.



8. In the Initial Context Factory field, verify or enter the name of the class (with full package) that is used to select the NIS provider.

9. In the Naming Service URL text field, verify or enter the NIS server URL. Valid formats include:

   ```
   nis://<hostname>/<domainname>
   nis:///<domainname>
   nis:/<domainname>
   nis:<domainname>
   ```

   The host name or IP address of the server `<hostname>` that is offering the NIS service to a domain `<domainname>` must be defined, otherwise the process may fail.

10. In the first User Service Name field, you may enter the name for the map list that contains group security information, in which group name is the key.

11. In the first User Attribute Name field, enter member identification number for the established domain.

12. In the second User Service Name field, you may enter the name for the map list that contains password information, in which the user name is the key.

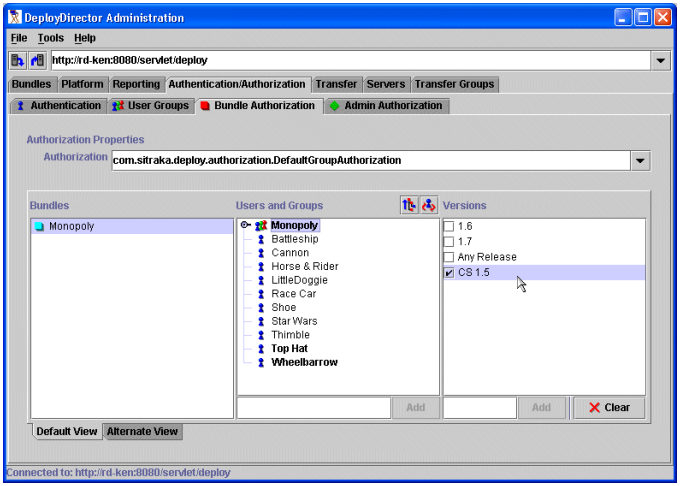
13. In the second User Attribute Name field, enter the group identification number in the established domain.

    Once the correct NIS information has been verified, you may want to review the contents of your server-side data file with the Default Authorization editor.

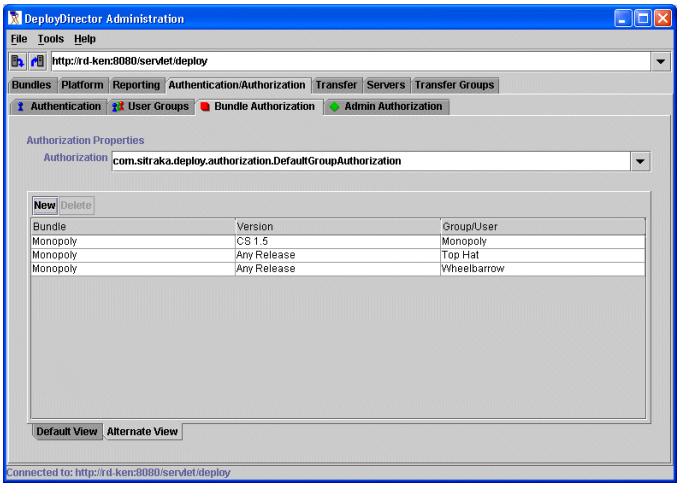
14. Click the Bundle Authorization child tab.

15. From the Authorization drop-down list, select the (`com.sitraka.deploy.authorization.`) `DefaultGroupAuthorization` class.

    Selecting this class shows the status of the server-side group authorization data file, which is referenced by DeployDirector during group authorization. This view shows associations between relevant bundles (i.e. those that use the `DefaultAuthorization` class), groups and users, and bundle versions.



The Alternate View displays the contents of the group authorization data file.



16. Review and if required, edit the groups or users that are authorized to use the bundle version.

    You can learn more about authorizing users or groups to particular bundles in End-User and Administrator Authentication Lists in Chapter 9.

17. Continue to configure your bundle, then commit these changes to the server.

An Overview of Security in DeployDirector

DeployDirector ensures the security of data during transmission, as well as the integrity of transmitted data. Transmission security is implemented through Netscape's SSL (Secure Sockets Layer), while data validation is based on an MD5 hash code. SSL implementation is flexible, in that you can choose from many third party SSL solutions or customize your own. This flexibility, combined with the robustness of these encryption schemes, ensures secure data transmission with DeployDirector.

About SSL and Symmetric Encryption

SSL: The SSL (Secure Sockets Layer) protocol is commonly used for secure client-server communication on the Internet, and ensures privacy when SSL-compliant machines communicate with each other. This encryption method uses a public and symmetric key: when clients and servers send data to each other, encryption is performed based on an encryption scheme provided by a public key (i.e. the public key is used to securely exchange a symmetric key).

If a client is sending data to a server, when a client establishes a connection, the server sends its public key back to the client. The public key is used by the client to encrypt data, which is then sent back to the server. The server then decrypts this data with its complimentary private key (which the server never sends to a client).

Symmetric encryption: The use of a single, common key in a key exchange between client and server is the first step in symmetric encryption. This encryption method involves the use of one secret key, which is used for encrypting and decrypting. During a transaction between client and server, the secret key is sent to the recipient of encrypted data. This encryption method is less secure with Internet transactions, where many users are anonymous. If any party acquires a copy of the secret key, all transmissions can be decrypted. However, encrypting data with a symmetric encryption method is significantly faster (up to ten times faster on the same data) than SSL encryption.

DeployDirector achieves a balance between client-server security and encryption speed by combining these two encryption approaches. The result is a secure and fast bundle transmission.

How Encryption Is Implemented in DeployDirector

DeployDirector uses public key encryption, authenticated with certificates, to encrypt a secret symmetric encryption key.

When a server receives a request from a client, it sends a public key to that client. The client uses the public key to encrypt a randomly generated symmetric encryption key, then (securely) sends this back to the server. Once this secret key has been sent back to the server, the server uses this secret key to encrypt the actual data, which is sent to the client.

Nesting symmetric encryption in a public key maximizes both security and speed. Since the secret key takes less time than a public key to encrypt data, it is used to encrypt the actual data. The public key, being more suitable for Internet transmissions, is used to transfer the secret key.

When a server is running in SSL mode, it can only communicate through the SSL protocol. Thus, an SSL connection can only occur between an SSL-enabled client and SSL-enabled server.

SSL Support with DeployDirector

DeployDirector does not have built-in SSL support; instead, the use of SSL in both the CAM and SAM is facilitated by an interface into which you can plug a customized SSL implementation, or an implementation from a third party library. DeployDirector uses or allows the use of SSL. As such, it includes a support code for SSL implementation. Some default support code for some more common third party libraries are also included. Implementing SSL support is covered in the next section of this chapter.

DeployDirector's open SSL interface offers more flexibility over built-in SSL support. Each organization's choice of which SSL implementation to use addresses two concerns.

The export of the encryption technology required by SSL is controlled by most governments. If SSL encryption technology was included in DeployDirector, a government export permit would most likely be required. Consequently, in order to comply with export regulations, a reduction in the key length of the encryption could be the result. Conversely, there exist countries that control the import of encryption technology.

Every organization's deployment situation is unique, and there exist some organizations for whom security is extremely important. Having an open SSL interface allows an organization to use a custom encryption scheme that addresses its specific security needs, or allows the use of a trusted third party library.

SSL Notes and Encryption Resources

While a reduced key length could result in reduced security, this may not pose any critical problems, unless your organization's data transfers require stringent security (e.g. online financial transactions). Currently, most export restrictions focus on the length of the key used to encrypt and decrypt each transmission. While transmissions encrypted with shorter keys are easier to crack by a socially retarded, but technically skilled eavesdropper, "easier" is a relative term. To illustrate this point, the RSA Labs FAQ (`http://www.rsasecurity.com/rsalabs/faq/`) states:

> While exhaustive search of DES's 56-bit key space would take hundreds of years on the fastest general purpose computer available today, the growth of the Internet has made it possible to utilize thousands of such machines in a distributed search by partitioning the key space and distributing small portions to each of a large number of computers. Recently, a group called distributed.net solved RSA's DES Challenge II, using an estimated 50,000 processors to search 85% of the possible keys, in 39 days.

56-bit DES encryption is now considered exportable from the United States and Canada. Whether or not this level of risk is acceptable is something each customer must decide for themselves.

As a counterpoint, also from the RSA Labs FAQ:

> Absent a major breakthrough in quantum computing (see Question 7.17), it is unlikely that 128-bit keys, such as those used in IDEA (see Question 3.6.7) or RC5-32/12/16 (see Question 3.6.4), will be broken by exhaustive search in the foreseeable future.

128-bit RC5 and IDEA encryption is not currently exportable from the United States without approval of the US Bureau of Export Administration (`http://www.bxa.doc.gov/`).

Aside from RSA Lab's introduction to cryptography, "Frequently Asked Questions about Today's Cryptography", available at `http://www.rsasecurity.com/rsalabs/faq/`, there are other sites that can help if you need to get better acquainted with some of the encryption technology used in DeployDirector.

You will find an introduction to SSL encryption at `http://developer.netscape.com/docs/manuals/security/sslin/contents.htm`.

A more general SSL technical manual area exists at `http://developer.netscape.com/docs/manuals/index.html?content=security.html`.

Additionally, you can also refer an SSL discussion FAQ at `http://www.faqs.org/faqs/computer-security/ssl-talk-faq/`.

In late 1999, SSL version 3.0 was renamed to TLS (Transport Layer Security) version 1.0, as the standard is now under the control of the IETF (Internet Engineering Task Force). The proposed TLS 1.0 standard can be found at `ftp://ftp.isi.edu/in-notes/frc2246.txt`.

DeployDirector's SSL Components

SSLFactory Method

The ability to plug in a third party of customized SSL implementation centers around DeployDirector's `com.sitraka.deploy.SSLFactory` interface. Implementing this interface with a class that bridges an SSL library with DD enables SSL encryption.

The class you use depends on the SSL library you use. DeployDirector includes several support classes for some more common third party SSL libraries.

Default SSL Implementations

The included SSL support classes that implement `com.sitraka.deploy.SSLFactory` support specific SSL extensions or libraries. These classes are located in `com.sitraka.deploy.ssl`.

Class	Name and Distributor	Where it can be obtained
JSSE.java	JSSE Java Secure Sockets Extensions Sun Microsystems	From Sun Microsystem's Javasoft Web site: http://java.sun.com/products/jsse/
IAIK.java	iSaSiLk SSL package IAIK or Entrust	From the IAIK-Java Group web site: http://jcewww.iaik.tu-graz.ac.at.
SSLJ.java	SSL-J RSA	RSA Security Web site: http://www.rsasecurity.com/products.bsafe/sslj.html.

If you are not using one of these, for all intents and purposes, you are using a customized one. More information about these are in the next section.

Proxies, Socks and Firewalls

When DeployDirector attempts to establish a connection for bundle deployment, it creates the raw connection itself. In doing so, an SSL handshake is invoked only after the connection procedure goes through SOCKS or http proxies. (Any application which needs to communicate through a proxy has to negotiate with the proxy first before continuing through the firewall.)

Since DeployDirector creates the connection, the SSL implementation must have a constructor that can accept an established socket. In light of this, three possibilities exist:

- The SSL implementation's constructor can accept an established socket: No extra steps need to be taken.
- Use the `startSSLHandshake()` method: If the SSL package does not accept an established socket, you can use this method to manually initiate the handshake process
- Use another or modify your SSL package: If the SSL package being used does not accept an established socket, and does not allow you to use the `startSSLHandshake()` method, the deployment connection will not be able to go through proxies.

Setting DD Encryption

The two pieces required are an SSL package, and the class that bridges that package with DeployDirector. The bridging class must:

- implement the `com.sitraka.deploy.SSLFactory`
- provide a no argument constructor
- be available in the CLASSPATH of component that requires it.

  For the server/servlet (SAM) this means that the SSL and related support classes must either be added to the `ddsam.jar` file or to the CLASSPATH of the servlet engine.

  For the client (CAM), the support classes must either be added to the `ddcam.jar` file or listed as part of your application's files and be added to the CLASSPATH object in the XML for each version of the application. If the SSL library requires native code libraries, then you will have to ensure that these are available as well. Keep in mind that native code is only supported in client applications under JDK 1.2. If JDK 1.1 is used, you must put the native code in the CAM JAR.

If Your SSL Library Is Not Supported

Your first step is to implement the `com.sitraka.deploy.SSLFactory` interface. Details on the implementation of this interface can be found in the Javadoc for that class. You will then need to add this class to the 3rd party library's JAR file. Example implementations can be found in the SDK along with the corresponding source code. Your SSL library can now be supported by DeployDirector.

If Your SSL Library Is Supported

If your SSL library is one of those listed under the 'Default Implementations', then you only need to ensure that the required classes and JAR files are available for the client or the server, and provide the class name of the implementing class.

For the client, add the libraries with the CAM. This is done by creating a new version of the CAM bundle that includes the SSL libraries. Make sure that you add the SSL jar files to the CLASSPATH object of the CAM bundle. Then you will have to provide the name of the implementing class. This is done on a per application basis by setting adding a `SYSTEMPROP` entry as follows:

```
deploy.http.sslsocketfactory = <desired SSL class>
```

For the server, either combine the ddsam.jar with the 3rd party libraries or add the 3rd party libraries to the CLASSPATH of the server. Then set the property `deploy.http.sslsocketfactory` in `server.properties` to the name of the class implementing the `com.sitraka.deploy.SSLFactory` interface.

Indicating which SSL security class to use with DeployDirector

1. In the Remote Administrator, navigate to the Server: Server Configuration: Miscellaneous Properties page.
2. In the deploy.http.sslsocketfactory field, enter the name of the class that implements the `com.sitraka.deploy.SSLFactory` interface, using the full package.
3. In the deploy.http.timeout field, enter the value that represents the time the client machine will wait for a server connection to be established before the connection is considered to have failed. (Please refer to Administration Tool Date and Time Entry Formats in Chapter 2, Introduction.)
4. Click Update Configuration to update the server to which you are connected with this change.

Overview of Data Validation

Since data integrity is vulnerable to both tampering as well as transmission errors, validity is ensured through the use of an MD5 hash code. The source of this binary code is the entire deployment bundle. This code, which is then converted to a string, is created at both ends of the deployment chain: the SAM creates it and attaches it to the bundle that is deployed; after receiving the bundle, the CAM also creates a code based on the bundle it received. This new code is compared to the one attached to the original bundle. A match offers reassurance that what was received is identical to what was deployed.

DeployDirector's security component consists of an interface with which any customized, or third party SSL implementation can be used. This offers the most flexibility for organizations, since security needs are often dependent on the scope of the deployment, as well as the type of data being transmitted. SSL support code is provided for some common third party libraries.

Chapter 7
Configuring Bundle Update Policies

Application updates can be deployed as new bundle versions. DeployDirector can handle updates in a conventional way by pushing out new versions to users. Alternatively, more control can also be given to the client-side, allowing end users to initiate updates.

The Client-Side Update Process

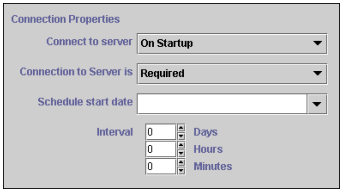
Whenever a CAM connects to a server (whether that connection is initiated by the user or is automatically established), the CAM asks the server to see if an update can take place.

When a new bundle version is available in the vault, the conditions under which a client machine's CAM retrieves and installs that bundle version can vary. While it may be typical that end users always work with the most recent bundle version (and have little choice in this matter), this does not have to be the case. Bundles can be configured to give end users more control over when they upgrade to a new version and when they connect to the server to determine if new versions exist.

As was previously mentioned, a CAM determines how bundle updates for a client machine are handled by reading specific properties of the bundle currently in use, and those of newer versions detected on the server. (The CAM asks the server for the update policy of the newer version. The policies that affect CAM update actions are Connection and Update.

Valid Connection Policy Settings

In the Administration Tool, selecting a bundle's Connection node reveals its connection policy:



Connect To Server indicates when the running bundle should attempt to connect to the deployment server. The Connection to Server Is setting indicates whether or not a connection to the server is required for the bundle to start up, or continue running. These settings, supported by Schedule settings if needed, constitute your bundle's connection policy.

The following outlines which Connection property settings are meant to work with others.

Connect to Server	Connection to Server Is	Schedule Information Required?
On Startup	Required / Preferred	no
Scheduled	Required / Preferred	yes
User Initiated	Preferred	no

Setting Bundle Connection Policies

Setting these properties determines when the client machine connects to the server, and if the user has control over when their machine connects to the server. This is important, since a connection to the server is required in order for the CAM to determine (by asking the SAM) whether newer versions exist.

Connection policy can be set for bundle versions in the Administration Tool.

Setting automatic mandatory client connections to a server on startup

1. Select the bundle version's Connection node to reveal its policy editor in the right pane.
2. From the Connect to Server drop-down list, select On Startup.
3. From the Connection to Server drop-down list, select Required.

4. Continue to configure your bundle, or commit these changes to the server.

Scheduling client connections to a server

1. Select the bundle version's Connection node to reveal its policy editor in the right pane.
2. From the Connect to Server combo box, select Scheduled.
3. From the Connection to Server combo box, select Required.
4. Click the combo box arrow in the Schedule Start Date field to reveal the calendar popup.
5. From the calendar popup, select the time and date on which you would like the first scheduled client connection to the server to occur.

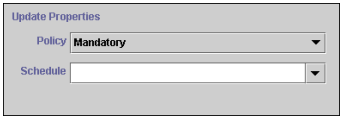
   Clicking a day on the calendar completes the selection process. The chosen schedule information is displayed in the Schedule Start Date field.
6. For the Interval property (specifically, the Days, Hours and Minutes spin boxes), indicate how often this scheduled connection will be re-established in the future.
7. Continue to configure other properties for your bundle version, or commit these changes to the server.

Giving the user control over server connections

1. Select the bundle version's Connection node to reveal its policy editor in the right pane.
2. From the Connect to Server combo box, select User Initiated.
3. From the Connection to Server combo box, select Preferred.
4. Continue to configure other bundle properties, or commit these changes to the server.

Setting Bundle Update Policies

A bundle's Update node, when selected in the Administration Tool, also reveals its own policy editor:



When a CAM connects to a server, and determines that a newer bundle version exists, the Update policy set on this newer version determines whether or not an update actually occurs, or will occur in the future.

An update policy can be set for bundle versions in the Administration Tool.

Making a bundle version a mandatory update

1. Select the bundle version's Update node to reveal its policy editor in the right pane.
2. From the Policy combo box, select either Mandatory or Scheduled Mandatory.
3. If you selected Scheduled Mandatory in the last step, click the combo box arrow in the Schedule field to reveal the calendar popup.
4. In the calendar popup, select the time and date on which you would like the bundle version to become a mandatory update.

   Clicking a day on the calendar completes the selection process. The chosen schedule information is displayed in the Schedule field.
5. Continue to configure other bundle properties, or commit these changes to the server.

Making a bundle version an optional update

1. Select the bundle version's Update node to reveal its policy editor in the right pane.
2. From the Policy combo box, select Optional.
3. Continue to configure other bundle properties, or commit these changes to the server.

The Connection and Update Options from the User's Perspective

When the end user runs a deployed application, and the CAM detects a newer version on the server, the feedback the user receives depends on how update policy has been defined.

Connecting to the server, if scheduled or performed whenever the application is started, is transparent to the user. However, upon connection, when the CAM detects a newer bundle version, that bundle's Update properties will result in the user being presented with a number of choices. The following table outlines the different update queries that are presented to end users, and the outcomes of their choices.

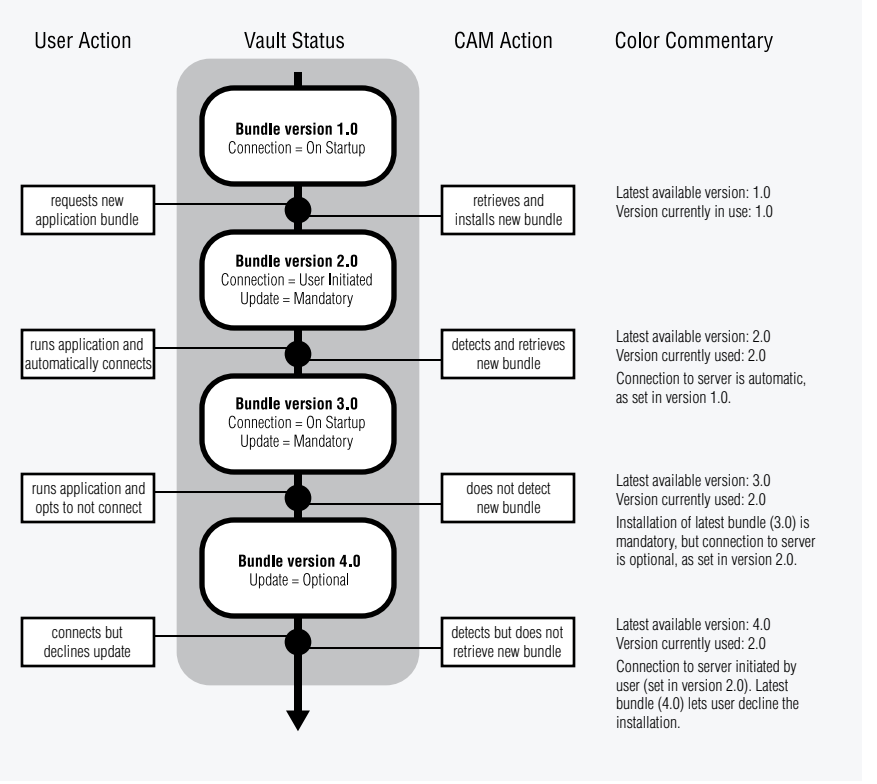
Bundle Update Policy	User Options	Result
Mandatory	update now	new bundle installs
	do not update	application terminates
Optional	update now	new bundle installs
	update in the background	new bundle brought down in the background; user prompted to update once all files present and application has been exited
	update later	does not install, currently installed application starts
	don't ask me again	does not install; currently installed application starts; end user is not prompted again until the CAM detects the next version of the bundle is present in the vault and the end user starts up the application
Scheduled Mandatory (before date)	update now	installs in the foreground
	update in the background	new bundle installs in the background
	update later	new bundle is not installed, currently installed application starts
Scheduled Mandatory (on or after date)	update now	new bundle installs
	do not update	application terminates

Examined separately, a bundle's Connection and Update policies seem pretty unspectacular. However, it is the combined effect of these policies, across multiple bundle versions, that requires careful consideration. The next three sections illustrate some of the effects of these properties.

CAM Update Example: Dependencies Between Bundle Versions

As a general rule, when one is using a bundle and newer bundle versions exist, *it is the current bundle's policy that determine how the connection to the server occurs*, and *it is the latest bundle's policy that determine how updates are handled.*

As such, when setting update policy for a new bundle version that the end user does not yet have, it is important to make sure you are aware of the Connection policy of the bundle with which the user is currently working. Conversely, when setting connection policies on a bundle version, it should fit into the overall update policy that you will implement for all future bundle versions. The following example illustrates this concept.

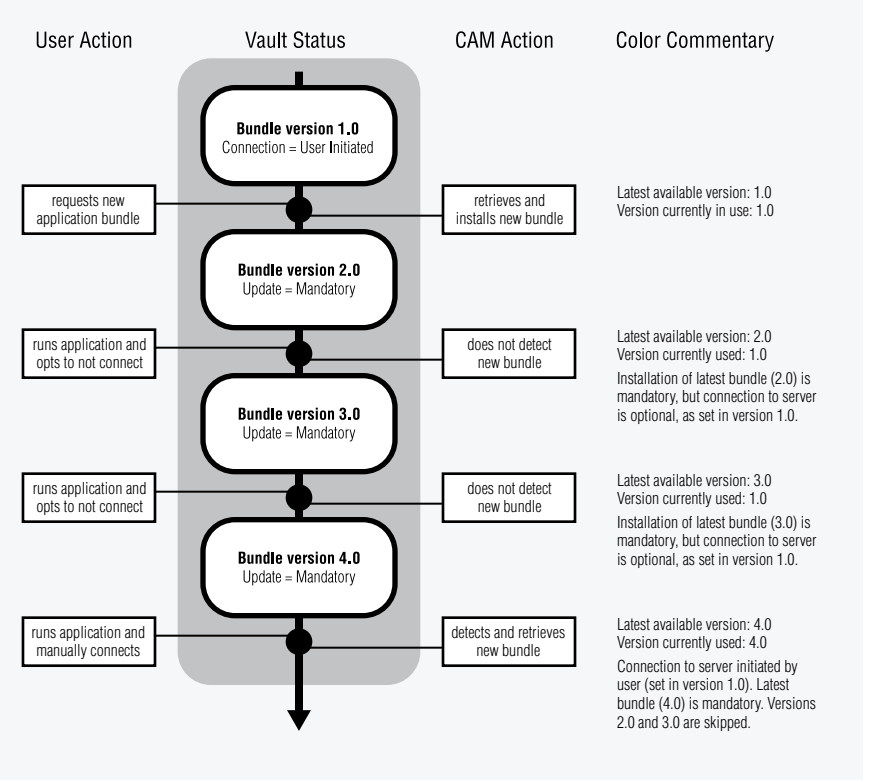


CAM Update Example: Effects of the Connection Policy

If your organization's deployment policy (whether written or unwritten) requires end users to work with the latest available bundles, it is important that their client machine establishes a connection with the server as often as you require.

A client machine can connect every time its user runs the deployed application, or to minimize network traffic their machine can connect at set intervals. This depends entirely on how the current bundle's Connection policy has been set. If your end users are not required to always work with the latest available bundle version, the choice to connect to the server can be given to the user.

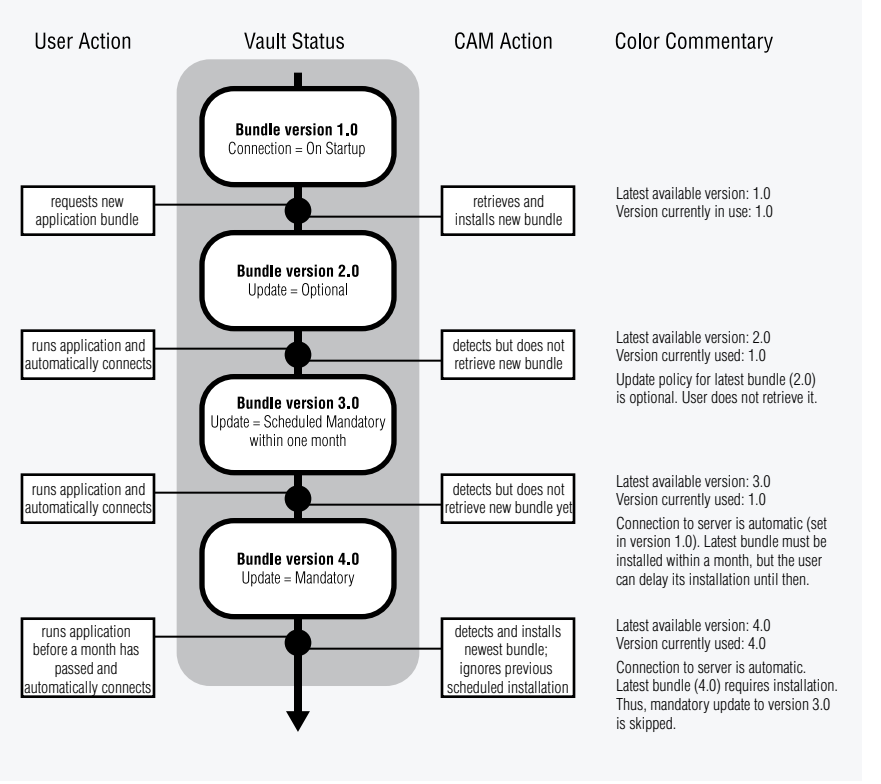
It is important to be aware of the effects of a bundle's Connection properties. The following example illustrates the result of a more liberal bundle connection policy, when combined with the (in)actions of a jaunty end user.



CAM Update Example: Effects of the Update Policy

Similar to the previous example, if your organization's deployment policy requires that end users work with the latest available bundles, it is important that bundle Update policy are properly set.

When a client machine connects to a server, and new bundles are detected by the CAM, the new bundle's update policy can be set as mandatory, mandatory by a specified date, or optional. It is important to be aware of the effects of these choices, and the following example illustrates the result of a mixed update policy across multiple bundle versions.



Chapter 8

Preparing Bundles and Servers for Deployment

Once your bundle has been fully configured, you need to prepare it for deployment. This means adding the new bundle or bundle version to your server's vault, and possibly transferring it to other servers for deployment. If the bundle is particularly large, you can create an installation CD for a manual installation on client machines (followed by deployed updates in the future).

Committing a Bundle to the Vault

Bundles that have been completely configured need to be uploaded to the vault. In the Administration Tool, clicking File > Update Server sends all uncommitted bundle versions to the server to which the Tool is connected. You can either connect and upload bundles to a production server that is part of a cluster (where it will be replicated to all others in the cluster), or to a server from which you can manually send the bundles to a transfer group.

Once a bundle has been committed (i.e. it has been uploaded to the server), it can then be deployed by that server and any other servers that are part of its cluster.

Please refer to Making Changes to the Vault in Chapter 4, Adding Bundles and Defining Bundle Content for information on other vault-related actions in the Administration Tool.

Preparing Bundles for Manual CD Installations

There are two ways of installing an application on the client side: from a CD-ROM or via Web browser. Both methods offer certain advantages. Installation of applications via a Web browser simplifies the process on the client side. It removes the need for a manual installation which can be time consuming and tiresome, especially if there is a large number of clients. Typically, administrators upload new bundles to a deployment server, and the client-side users then download it across your network. (Information on configuring bundles for installation from a Web browser can be found in Chapter 5, Configuring Bundle Installation Properties.)

If an application is large (resulting in a large bundle) or there are concerns regarding the availability of bandwidth, your organization may prefer to initially install the application on the client side via CD. After this initial manual step, the bundles are then managed remotely by DeployDirector and the end users receive updated bundle versions from the server across the network. DeployDirector provides class-level differencing during application updating which mitigates any subsequent concerns about the availability of bandwidth.

This installation option offers more flexibility to your organization as well as to Independent Software Vendors (ISVs), who can package their application with DeployDirector. The end user would simply insert the CD, and choose the applications to be installed from a list of options presented in a DeployDirector Installer window. Later, such applications can be easily maintained and updated by the network administrator using DeployDirector.

An Overview of DARs

The Deploy Archive (DAR) file format is analogous to the JAR format. It has been created to encapsulate bundles for transport. This format preserves the file names, directory information, and file attributes of all files contained in a bundle. DARs can be generated to simplify the transfer of bundles from a development server to a production server, or between two servers that are not part of the same cluster.

DARs are used to package bundles that will be installed on the client side from a CD and later managed by DeployDirector. First, the bundles to be distributed to the client desktops are encapsulated within DARs using the Administration Tool or the DAR tool. Second, the DARs are placed in the `vault` directory on the CD. Though the initial installation on the client side is manual, the bundles are later managed automatically using DeployDirector.

The DARs should be named to adhere to the ISO9660 convention (at most, eight characters long, followed by a file extension of maximum three characters) to support the CD installation option described above.

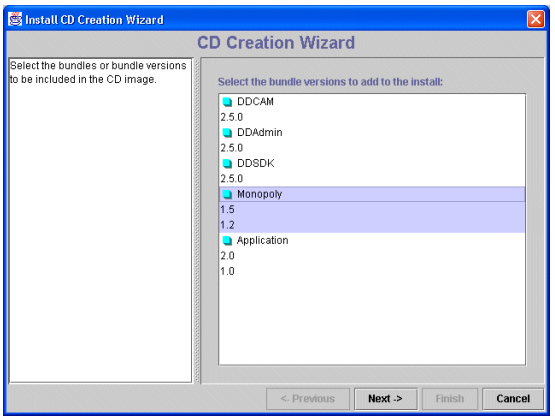
The structure of the DARs follows the example of the JAR format: the bundle files and a META-INF subdirectory. The files describing the bundle, like `version.xml` and `bundlename.txt`, are stored in the META-INF directory within the archive.

DARs can be easily created in the Administration Tool by selecting the bundle version you wish to export, then clicking File > Export Bundle. The Administration Tool accommodates DAR import and export, easily allowing you to prepare bundles for transport on installation CDs. (Alternatively, DARs can also be exported in the Remote Administrator, on the Bundle: Export DAR page.)

DeployDirector also includes command line DAR tools that have a much richer feature set, and can also work with WAR files. The presence of the DAR command line tool facilitates automated bundle packaging and distribution through scripts, as well as WAR and DAR conversion. Please see Using the DAR Command Line Tool found later in this chapter for extensive information about the tool's commands and options.

Setting Up an Installation CD

The easiest way to create an installation CD is to use the CD Creation Wizard. This can be accessed by clicking Edit > CD Wizard in the Administration Tool.



Using the Wizard, you are walked through the essential steps needed to create a CD-based installation. However, you can maintain greater control by performing this process manually.

The following procedures outline the process of packaging your applications with DeployDirector to create an installation CD. This process includes exporting your bundles as DARs, moving necessary DeployDirector files over to the CD burn area, and burning the CD.

Important: If a higher bundle version exists that is configured as a mandatory update, the CD-installed bundle will update as soon as it is run.

Saving bundles and the CAM as DARs

1. Ensure that the bundles you wish to put on an installation CD have been configured properly, and have been uploaded to the server.
2. In the Administration Tool, select the bundle version you want to put on an installation CD.
3. Click File > Export Bundle.

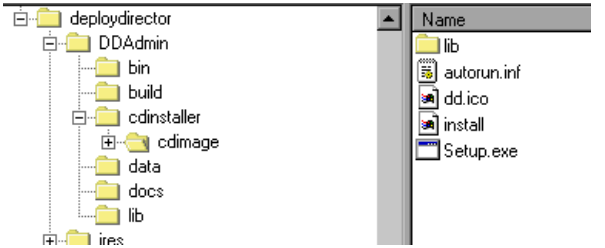
   As an alternative, you can create a DAR using the command line tool included in the Administration Tool distribution (DDAdmin) in the `<installpath>/deploydirector/DDAdmin/bin` directory.
4. In the Save dialog box, navigate to the directory where you are assembling files to be burned onto CD.
5. Enter the name of the DAR, followed by the `.dar` extension, and click Save.

   You have now created and saved the DAR for the version of the bundle which is destined for the installation CD.

   Similarly, create a DAR file for each bundle that is going to be distributed to the client desktops.
6. Select the current version of the DDCAM bundle.
7. Create a DAR of the DDCAM by clicking File > Export Bundle.
8. In the Save dialog box, navigate to the directory where the DARs were saved from previous steps, and save the DDCAM as `ddcam.dar`.

Assembling the necessary DeployDirector and DARs

1. When you first deployed the Administration Tool to your workstation, the following directory structure was installed:



   This entire `cdimage` directory must be copied to the CD-ROM, as described in the next step.

2. Set up the following directory structure for your CD-ROM.

```
target directory
    |--setup.exe
    |--autorun.inf
    |--dd.ico
    |--lib
    |  |--cdsetup.jar
    |--install
    |
    |--jres
    |  |--unix
    |     |--i386
    |        |--linux
    |           |--{manufacturer}
    |  |--unix
    |     |--sparc
    |        |--solaris
    |           |--{manufacturer}
    |  |--windows
    |     |--i386
    |        |--sun
    |           |--bin
    |           |--lib
    |
    |--deploydirector
    |  |--jres
    |  |--vault
    |  |--cluster.properties
    |  |--platform.xml
    |  |--server.properties
```

These files are copied over from the Administration Tool's `cdimage` directory.

The contents of the `jres` directory is dependent on your client machines. JREs can be included for the Windows, Linux, and Solaris platforms.

Please see Step 5 for more information about adding JREs to your installation CD.

The `deploydirector` directory and `/jres` and `/vault` subdirectories should be copied from the server after the bundles have been configured and the changes committed to the server.

3. Remove all files and folders from the `/vault` subdirectory.
4. Copy all of the DARs into the `/vault` directory.

   The `vault` directory should contain `ddcam.dar` as well as all the DARs you created in the previous procedure.
5. Copy the contents of the `/jres` directory to the `/deploydirector/jres` directory.
6. If you are targeting a Windows platform, ensure that you include the unarchived contents (including the `/lib` and `/bin` subdirectories and contents) of a Sun JRE 1.3 or greater. (To test whether the Windows `jres` directory has been set up properly, verify that the following command can be executed:
   `[target directory]/jres/windows/i386/sun/bin/javaw.exe`

A ZIP file containing Sun's JRE 1.3 is provided in the DeployDirector distribution under `<installpath>/DeployDirector/jres/windows/i386/sun/1.3.0/prebuilt.zip`. For convenience, you can extract this `prebuilt.zip` file into the `jres` directory.

If you are including JREs for Unix clients, ensure that all file names under the `/jres` subdirectory do *not* contain periods. Replace all periods with underscores (e.g. `/sun/1.2.2` becomes `/sun/1_2_2`).

If you wish to reduce the number of files that are burnt onto a CD, you can safely delete any JREs not required on the client side from the `[target directory]/deploydirector/jres` directory. The JREs should also be removed from the `platform.xml` file.

Including customized install event classes

If you are using a customized install event class for a bundle, it will already have been included in your DDCAM bundle. (Please refer to the section entitled Extending Installation Options with Custom Classes for more information about including these classes to the DDCAM bundle.)

For a DeployDirector CD installation, you will also have to include this class in the `cdsetup.jar` archive, which was copied over to a temporary area in a previous procedure.

1. Locate the `cdsetup.jar` in the `cdinstaller/cdimage/lib directory`.
2. Add the install event class to this archive.

Writing the image to a CD

1. You can test the CD Installer on your local drive before burning the directory structure to a CD-ROM. For the Windows platform, run the `setup.exe` file. For Unix platform, run the `install` script.
2. When configuring the settings for burning the CD, please ensure that you select 'File System: ISO9660' in the CD Layout Property dialog box. The burner will prompt you to rename all files containing two dots in the file name, such as `font.properties.*`. In order to ensure a smooth CD-burning session, please rename these files so that there is no more than one dot in the file name. In addition, make sure that the file name support option is set to 'long file names (maximum 30 characters)'. The CD-ROM should be burnt in one session, at the write speed of 1x (150 KB/sec).

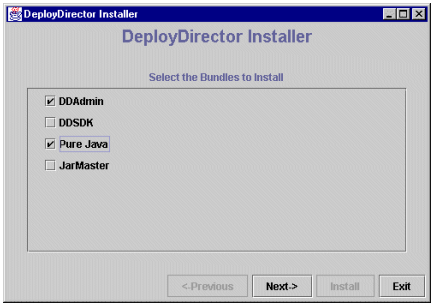
Installing an Application from a CD-ROM

1. Insert the CD into the CD-ROM drive.
2. Choose from the following options:
   - For Windows: If the auto-run feature does not initiate the installation process automatically, choose Start > Run. Browse to the directory containing the `setup.exe` file on your CD. Choose the `setup.exe` file. Click OK in the Run dialog box to begin the installation
   - For Unix: invoke the `install` script to begin the installation. Ensure that JDK 1.2.2 or above is already installed on the client machine.

   Note: In order to mount a CD in HP-UX, you will have to enter the following commands:
   `su` (This will put you in the "super user" mode. You will need to supply the root password.)
   `mkdir /cdrom` (You probably want to add this directory at the root of the file system.)
   `mount -F cdfs -o cdcase /dev/dsk/cdrom_device /cdrom` (where `cdrom_device` is listed in the output of the `ioscan -f -n` command)

   To unmount the CD, you will need to enter the following command:
   `umount /cdrom` (where `/cdrom` is the location where you mounted the CD).
3. Follow the on-screen instructions.

   If you are simultaneously installing several applications, you will be presented with a list of all available bundles archived in the DAR.



   You can select multiple applications from this list. The DeployDirector Installer will guide you through the installation process.

Using the DAR Command Line Tool

The DAR command line tool automates the process of generating DARs. The tool (`dar.jar` and `dar.bat`) can be found in the `<installpath>/deploydirector/DDAdmin/bin` directory within the DeployDirector Administration Tool distribution.

The use of the tool begins with the `dar` command, and follows this template:

```
dar <command> <options>
```

The tool augments the basic DAR import and export commands available in the Administration Tool, and also allows you to convert WARs to DARs, as well as create DARs from local files (i.e. without first creating bundles).

dar convert: conversion of a WAR file to a DAR file

The `dar convert` command is used to convert a WAR into a DAR, and uses the following options:

Option	Purpose:
`-d <dar name>` OR `--dar <dar name>`	Assigns a name to the DAR. If this option is not used, and the `--bundle` option is used, a `<bundlename>.dar` file is created. Otherwise, the base name of the WAR (i.e. the name without the `.war` extension) is used.
`-w <source war>` OR `--war <source war>`	Specifies the WAR that will be converted into a DAR. If this option is not specified, the tool will search in the current working directory for a WAR. If one is not found, the tool will terminate and report an error.
`-b <bundle name>` OR `--bundle <bundle name>`	Specifies the name of the bundle to be written into the DAR. If this option is not specified, the default bundle name is the base name of the WAR being converted. This argument is mandatory if the WAR is being read from standard input.
`-v <version name>` OR `--version <version name>`	Assigns a version name to the converted archive. If this option is not specified, the default version name is used (`1.0.0`).
`-x <template XML>` OR `--xml <template XML>`	Provides a path to a template `version.xml` file that is meant to be read and processed. This file can provide settings for standard DeployDirector bundle properties such as UpdatePolicy and Connection. If this option is not specified, the standard bundle property settings are used.

dar import: importing a WAR or DAR file to the server

The `dar import` command will take a specified DAR or WAR and upload it to a particular server, thus converting it into a DeployDirector bundle. The command uses the following options:

Option	Purpose:

`-d <dar name>`
OR
`--dar <dar name>`
	Indicates the name of the DAR to be imported or uploaded to the vault. Specifying a hyphen as an argument instead of a DAR name (i.e. "`-d -`") results in the DAR being read from `System.in`.
	
	Only one of `-d` and `-w` can be used.

`-w <war name>`
OR
`--war <war name>`
	Indicates the name of the WAR to be imported or uploaded to the vault. Specifying a hyphen as an argument instead of a DAR name (i.e. "`-w -`") results in the WAR being read from `System.in`.
	
	Only one of `-d` and `-w` can be used.

`-b <bundle name>`
OR
`--bundle <bundle name>`
	Specifies a bundle name for the WAR or DAR when it is sent to the server.
	
	This option is mandatory if a WAR is being imported, and read from standard input.

`-v <version name>`
OR
`--version <version name>`
	Assigns a bundle version name to the imported WAR or DAR. If this information is not available (e.g. the bundle is being read from a WAR) the default version name is used (`1.0.0`).

`-p <server password>`
OR
`--password <server password>`
	Specifies the administrator password to access the server.
	
	This option is mandatory.

`-U <admin name>`
OR
`--user <admin name>`
	Specifies the administrator user name to access the server. If no name is provided, the default (ddadmin) is used.

`-u <server URL>`
OR
`--url <server URL>`
	Specifies the URL at which the server can be found. This option should include the full access path to the server, including the root path of the servlet (e.g. `http://foo.bar.com:8080/servlet/deploy`).
	
	This option is mandatory.

Option	Purpose:

`-x <template XML>`
OR
`--xml <template XML>`
	Provides a path to a template `version.xml` file that is meant to be read and processed. This file can provide settings for standard DeployDirector bundle properties such as UpdatePolicy and Connection. If this option is not specified, the standard bundle property settings are used.

dar export: exporting a bundle from the server as a DAR

The `dar export` command will take a specified bundle and convert it to a DAR. The command uses the following options:

Option	Purpose:

`-d <dar name>`
OR
`--dar <dar name>`
	Indicates the DAR name that will be used with the downloaded bundle. If this option is not used, the bundle name will be assigned to the DAR.
	
	Output can be sent to `System.out` if a hyphen is used as an argument (i.e. "`-d -`").

`-b <bundle name>`
OR
`--bundle <bundle name>`
	Specifies the name of the bundle whose version is to be retrieved from the server.
	
	This option is mandatory.

`-v <version name>`
OR
`--version <version name>`
	Specifies which version of the bundle is to be retrieved from the server. If a version is not specified, the latest bundle version is selected.

`-p <server password>`
OR
`--password <server password>`
	Specifies the administrator password (default is `f3nd3r`) to access the server.
	
	This option is mandatory.

`-U <admin name>`
OR
`--user <admin name>`
	Specifies the administrator user name to access the server. If no name is provided, the default (ddadmin) is used.

`-u <server URL>`
OR
`--url <server URL>`
	Specifies the URL at which the server can be found. This option should include the full access path to the server, including the root path of the servlet (e.g. `http://foo.bar.com:8080/servlet/deploy`).
	
	This option is mandatory.

dar create: creating a DAR

The `dar create` command locally creates a DAR based on the settings and files specified in the command options, as well as the bundle settings outlined in the `version.xml` file. The command uses the following options:

Option	Purpose:

`-d <dar name>`
OR
`--dar <dar name>`
	Indicates the name given to the newly created DAR name that will be used with the created bundle. If this option is not used, the bundle name will be assigned to the DAR.
	
	Output can be sent to `System.out` if a hyphen is used as an argument (i.e. "`-d -`").

`-b <bundle name>`
OR
`--bundle <bundle name>`
	Specifies the name of the bundle for which the DAR will be created.
	
	This option is mandatory.

`-v <version name>`
OR
`--version <version name>`
	Specifies the version name of the newly created DAR. If this option is not provided, the default version name will be used (`1.0.0`).

`-x <template XML>`
OR
`--xml <template XML>`
	Provides a path to a template `version.xml` file that is meant to be read and processed. This file can provide settings for standard DeployDirector bundle properties such as UpdatePolicy and Connection. If this option is not specified, the standard bundle property settings are used.

`-C <root directory>`
OR
`--changedir <root directory>`
	Changes focus to the specified directory and will continue to process files. All files listed on the command line or in a file list with the `--filelist` option will come from this new directory.
	
	The current directory in which the `dar create` command is used is the default directory, and can be explicitly referenced with a period (i.e. "`-C .`")

`-p <platform name>`
OR
`--platform <platform name>`
	Specifies the location of the platform hierarchy for the files included with the following `--filelist` option, and the following `--classpath` option until the next `--platform` is reached in the string of `dar create` options.
	
	The top level platform is the initial default platform, or it may be explicitly referenced by using `all` (i.e. "`-p all`").

Option	Purpose:

`-f <list of files>`
OR
`--filelist <list of files>`
OR
`<file> [<file> ... ]`
	Specifies the files (listed directly on the command line) or points to a file that contains the list of files to be included (one file name per line). Specifying a hyphen as an argument instead of a file name (i.e. "`-f -`") results in the file name being read from `System.in`.
	
	Since multiple platform information can be provided in a single dar create command, the `--filelist` (along with the `--classpath`) option can appear multiple times in a command.

`-c <"bundle classpath">`
OR
`--classpath <"bundle classpath">`
	Provides a CLASSPATH for the bundle used to create the DAR. The argument used with this option should be bound by quotes to accommodate the use of semi-colons.
	
	If this option is not provided, then all JARs found are added to the CLASSPATH.
	
	Since multiple platform information can be provided in a single dar create command, the `--classpath` (along with the `--filelist`) option can appear multiple times in a command.

Chapter 9
End User and Administrator Access

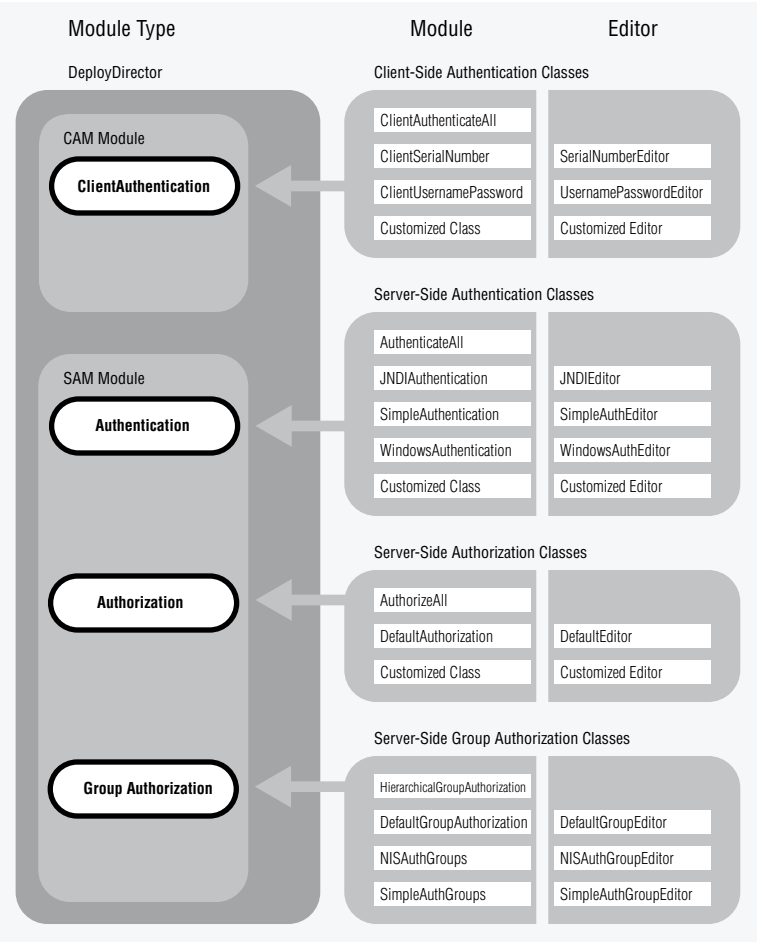
In Chapter 6, Configuring Bundle Runtime Properties, you learned how to configure bundle properties that affected how it behaved at runtime. This included how, at startup or installation time, a bundle would ensure the client-side user attempting to access it was permitted to do so. This functionality is handled by DeployDirector's set of authentication and authorization modules. In this chapter, you are given an in-depth explanation of these modules, and are shown how to use the Administration Tool to set up authorization lists for bundles. Additionally, you will learn about the use of the Administration Tool to define and manage administrator roles.

An Overview of User Authentication and Authorization

DeployDirector ensures that only certain users are allowed to access bundles on your organization's servers, and if required, that only certain subsets of users can access particular bundles or versions. Authentication and authorization comprise the two-punch process that ensures this occurs.

Authentication determines whether a user is who they claim to be, while authorization determines whether an authenticated user is permitted to install and run the particular bundle they are requesting.

Programmatically, authentication and authorization have been modularly implemented in DeployDirector, allowing you to plug in classes whose properties best match your deployment environment and needs. The following outline is meant to act as both an orientation to, and a reference for the programmatic side of authentication and authorization in DeployDirector:



The following sections describe the roles and properties of the module types, the modules that implement them, as well as the editors through which authentication or authorization information is received.

Authentication and Authorization Module Types

The three authentication/authorization module types, found in the `com.sitraka.deploy` package, are the code modules into which you can plug specific authentication and authorization classes. Since authentication and authorization is a process where exchanges occur between the CAM and SAM (i.e. the client and server sides), modules and classes are located in the appropriate area.

ClientAuthentication determines and manages the type of authentication information that is required, and how it is collected from the user. The type of class used also determines whether the information is stored on the client side.

Authentication manages the actual, server-side authentication of a user.

Authorization determines if the user (once they have been authenticated) has permission to access specific bundles and bundle versions. Authorization can be applied to both individual users, as well as defined authorization groups.

Client-Side Authentication Module and Editor Classes

The client-side authentication classes implement the `ClientAuthentication` interface. They reside on the client side (i.e. in the CAM), and collect the necessary information. This information is sent to the SAM, where it is handled by the server authentication module.

ClientAuthenticateAll is used when no user identification is required (i.e. the bundle is freely available to all who request it). Thus, it has no editor and contains no authentication information. It is typically used in conjunction with the `AuthenticateAll` server-side authentication class.

ClientSerialNumber is a serial number-based authentication class. The authentication information consists solely of a serial number. This class uses the `SerialNumberEditor` class to retrieve the serial number from the client.

ClientUsernamePassword is the commonly used client authentication class. It bases authentication information on a user name and password, and uses the `UsernamePasswordEditor` class to retrieve the user name and password combination from the client.

AbstractClientHTTPAuthentication is an abstract base class that implements basic HTTP authentication. This base class is used by both `ClientSerialNumber` and `ClientUsernamePassword`.

Client-side authentication editor classes work in tandem with their partner module classes by providing the GUI that gathers the authentication information from the user. Each editor was designed to be used with a specific module class.

SerialNumberEditor obtains a serial number from the client, and works with the `ClientSerialNumber` class. This editor typically runs in the bundle installation applet, and is used by the end user to input information.

UsernamePasswordEditor obtains a user name and password from the client, and works with the `ClientUsernamePassword` class. This editor typically runs in the bundle installation applet, and is used by the end user to input information.

Server-Side Authentication Module and Editor Classes

The server-side authentication classes implement the `Authentication` interface. They reside on the server side (i.e. in the SAM), and validate the authentication information (typically a user name and password) sent to it by the CAM-based authentication module.

AuthenticateAll is used when no user identification is required. Thus, it has no editor and receives no authentication information. This class is typically used in concert with the `ClientAuthenticateAll` client-side authentication class.

JNDIAuthentication is a JNDI-based authentication class. This class looks for a JNDI-accessible naming service that provides user name/password validation, which currently is only NIS (i.e. it validates user names and passwords against a Unix login). This class uses the `JNDIEditor` class to obtain configuration information.

Note: In order to use NIS authentication within the JNDI framework, two specialized JARs are required (`NIS.jar` and `ProviderUtil.jar`), which are available at Sun Microsystem's JNDI service provider's page at http://java.sun.com/products/jndi/serviceproviders.html. Downloading the NIS service provider provides a package that contains these JARs.

Once downloaded, your CLASSPATH needs to be modified to include these JARs. However, if you are using DeployDirector's standalone server, be sure the JARs are placed in the `deploydirector/lib` directory. When run, the standalone server will automatically add JARs found in that location to the CLASSPATH.

SimpleAuthentication is the most commonly used server-side authentication class. Its table-based authentication method validates user name/password information against a list of valid combinations provided in a data file. This class uses the `SimpleAuthEditor` class to get this information.

WindowsAuthentication is a Windows-based authentication class, which works similarly to `JNDIAuthentication`. This authentication module uses Windows 95/98/NT/2000 authentication to verify the user name and password combination provided by the client (i.e. it validates user names and passwords against a Windows login). The class uses the `WindowsAuthEditor` class to obtain its configuration information.

AbstractAuthentication is an abstract base class that implements basic server-side authentication. This base class is used by `JNDIAuthentication`, `SimpleAuthentication` and `WindowsAuthentication`.

Server authentication editors work together with their partner module classes by allowing the entry and edit of authentication information. This information can be a table of user authentication data (as with `SimpleAuthentication`) or it can be information required to set up a connection to authentication information (as with `JNDIAuthentication` and `WindowsAuthentication`). The server-side authentication editors run inside the DeployDirector Administration Tool, and each are used with a specific module class.

JNDIEditor lets system administrators configure the JNDI naming service look-up in the Administration Tool. This editor class is used with `JNDIAuthentication`.

SimpleAuthEditor lets system administrators enter and edit user name and password combinations. It is used with `SimpleAuthentication`.

WindowsAuthEditor lets system administrators configure Windows authentication look-up, and is used with `WindowsAuthentication`.

Authorization Module and Editor Classes

It is probably not surprising that authorization classes implement the `Authorization` interface. These classes determine whether a particular user has access to a particular version of a particular bundle. Since determining user access privileges to vault-based bundles is a server-side issue, these classes reside on the server side (i.e. in the SAM).

AuthorizeAll allows all users access to all bundles and bundle versions. This class is used when no user authorization is required. As such, this module class has no editor class, and since no authorization information exists, no data file is required.

DefaultAuthorization authorizes user/version/bundle combinations based on the contents of an authorization configuration file that stores valid combinations as comma-separated values. This class uses the `DefaultEditor` class in the Administration Tool and also works with the `Version` authorization support class.

Version handles the notion of a bundle version, and works with the `DefaultAuthorization` class. The `Version` class includes code that models any release, qualified releases and beta releases.

There is one authorization editor, which works in conjunction with the `DefaultAuthorization` module class.

DefaultEditor allows system administrators to configure `DefaultAuthorization` by allowing user/version/bundle entry input into the `.dat` file.

Group Authorization Module and Editor Classes

The following are special authorization classes that allow administrators to define groups of users that can be authorized to access specific bundles, instead of relying on the specific user-level settings that are used with regular authorization.

DefaultGroupAuthorization authorizes bundle/version/group combinations based on the contents of a data configuration file that stores valid combinations as comma-separated values. This class uses the `DefaultGroupEditor` class in the Administration Tool.

HierarchicalGroupAuthorization is similar to `DefaultGroupAuthorization`, but also takes into account the specificity of a user's presence in authorization lists (e.g. a user's individual authorization settings overrides those they have as part of an authorization group).

NISAuthGroups is an NIS-based group authorization class whose configuration information (maintained with the `JNDIAuthGroupEditor`) identifies the NIS server that is used to retrieved authorized users.

Note: In order to use NIS authentication within the JNDI framework, two specialized JARs are required (`NIS.jar` and `ProviderUtil.jar`), which are available at Sun Microsystem's JNDI service provider's page at http://java.sun.com/products/jndi/serviceproviders.html. Downloading the NIS service provider provides a package that contains these JARs.

Once downloaded, your CLASSPATH needs to be modified to include these JARs. However, if you are using DeployDirector's standalone server, be sure the JARs are placed in the `deploydirector/lib` directory. When run, the standalone server will automatically add JARs found in that location to the CLASSPATH.

SimpleAuthGroups facilitates the creation and maintenance of authorization groups, by using a data file that is managed using the `SimpleAuthGroupEditor` in the Administration Tool.

Group authorization editors work together with their sister module classes by allowing the entry and editing of information that defines authorization groups (`SimpleAuthGroups`), associates bundles with established authorization groups (`DefaultGroupAuthorization`), or information that is required to set up a connection to authorization information (`JNDIAuthGroupEditor`).

DefaultGroupEditor facilitates the creation and maintenance of authorization groups for system administrators. Working with the `DefaultGroupAuthorization` class, group names and the members of which they are comprised are defined here.

NISAuthGroupEditor lets system administrators configure the NIS naming service look-up in the Administration Tool for group authorization. This editor class is used with `NISAuthGroups`.

SimpleAuthGroupEditor allows the maintenance and creation of authorization groups. Group names and their respective members (separated by commas) are inputted in this editor.

Authorization Behavior and Allowable Bundle Version Names

Some of the authorization classes mentioned in the previous two sections (specifically, `DefaultAuthorization`, `DefaultGroupAuthorization`, and `HierarchicalGroupAuthorization`), affect how a bundle's versions are named. When you configure a bundle version to use any of these authorization classes, it is important that its version name conforms to an `x.y.z` style format, where every variable is a whole number, and in which there can be a maximum of three. When using these authorization classes, valid bundle version names include:

```
1
1.4
1.4.999999
```

The three authorization classes reference the same class file, which processes version names that strictly conform to this format. (For those who are working with the DeployDirector SDK, look for the Version class, found in `com.sitraka.deploy.authorization`).

This naming constraint enables more flexible authorization statements. For example, authorizing a user for a more general version name (e.g. `1.2`) means that user is authorized to access any version name that begins with that name (e.g. they would be automatically authorized to access `1.2.5` and `1.2.10`).

Please refer to Adding and Removing Bundles in Chapter 4 for information on adding and naming bundle versions.

Authentication and Authorization Configuration Files

Typically, any module class you use will require a configuration file, to which the class is pointed during authentication or authorization. The type of information contained in the file is specifically related to the module class it accompanies. Thus every class that requires a data configuration file has its own specific version.

Simpler configuration files (e.g. `SimpleAuthentication` and `DefaultAuthorization`) can contain raw information, while other files (e.g. `JNDIAuthentication` and `WindowsAuthentication`) contain information that redirect to a source that contains validation information. The file name is based on the module class name. For example:

Module Class Name	Accompanying Configuration File Name (in com.sitraka.deploy.)
DefaultAuthorization	authorization.DefaultAuthorization.authorization.dat
DefaultGroupAuthorization	authorization.DefaultGroupAuthorization.authorization.dat
JNDIAuthentication	authentication.JNDIAuthentication.authentication.dat
JNDIAuthGroups	authentication.JNDIAuthGroups.authentication.dat
SimpleAuthentication	authentication.SimpleAuthentication.authentication.dat
SimpleAuthGroups	authentication.SimpleAuthGroups.authentication.dat
WindowsAuthentication	authentication.WindowsAuthentication.authentication.dat

These files are found in the `/auth` directory, and are first created when any authentication or authorization information are initially uploaded to the server.

End-User and Administrator Authentication Lists

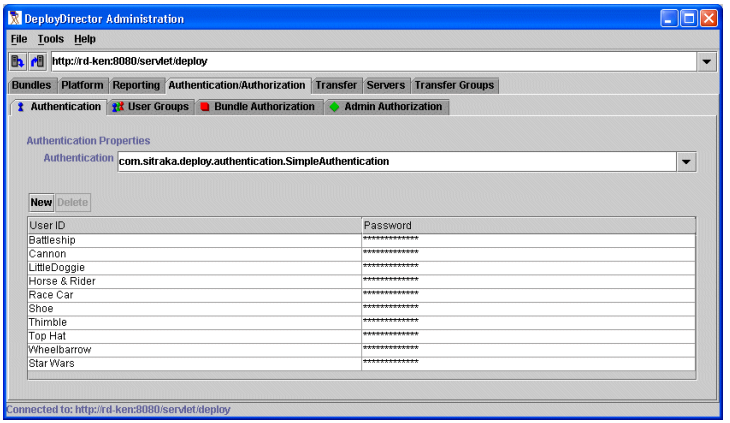
DeployDirector uses authentication whenever client-side users try to download or run a bundle, or when an administrator attempts to gain access to a server via the Administration Tool or the Remote Administrator. In both cases, DeployDirector maintains server-side data files for each of the authentication modules that may be used. These are the basic `SimpleAuthentication` (for individual users), and `SimpleAuthGroups` (for groups of users) modules. Other authentication modules exist, but these are centered around the use of your organization's JNDI directories or Windows login information. As such the data files associated with these modules do not include user data; the information they contain possess information on the NIS server or Windows domain.

Unlike company directories, the `SimpleAuthentication` and `SimpleAuthGroups` modules' data files are under complete control by DeployDirector. This means their maintenance is performed via the Administration Tool. It is important that the contents of these files are kept up to date, since they are displayed in the authorization editors for bundles that use these modules. Authorizing users that have not been authenticated is obviously not recommended.

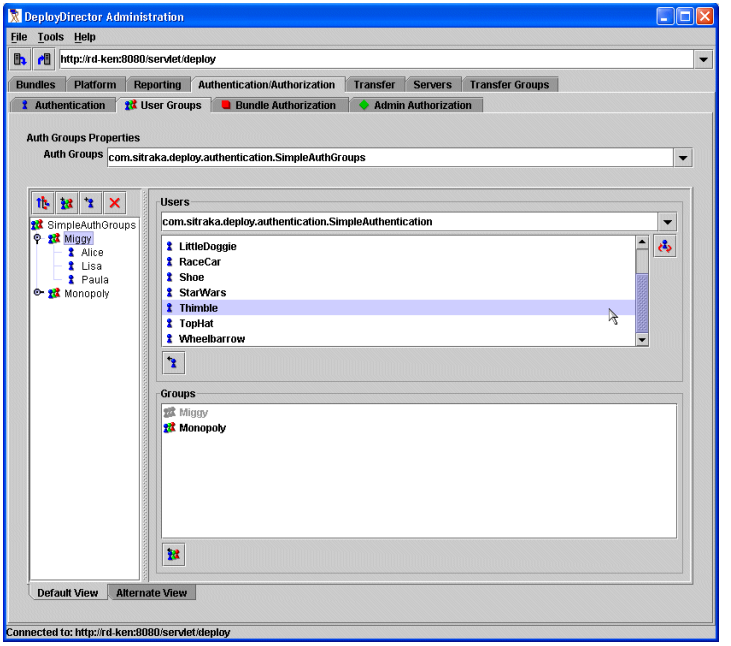
Regardless of whether you use server data files or directories to populate Users and Groups lists with authenticated members, you need to understand how these lists are populated. This is dependent on the combination of authentication and authorization classes chosen in the bundle configuration. Please refer to the next section, Viewing End-User Bundle Associations, for more information about the actual process of authorizing users.

Viewing Authentication Lists

You can view the current list of authenticated individual users (i.e. the contents of the server file associated with the `SimpleAuthentication` module, with which the primary components are user names and passwords) by clicking the Authentication/Authorization tab in the Administration Tool, then clicking the Authentication child tab.



You can view authentication groups and their respective sets of members (i.e. the contents of the server file associated with the `SimpleAuthGroups` module) by clicking the User Groups child tab, and selecting the `SimpleAuthGroups` class in the Auth Groups drop-down list.



This group authentication editor allows you to work in two modes: the Default View and Alternate View. With either view, you can create new groups, and add groups or users to existing groups. Whenever changes are made to either file, you must click the Update Server icon (or click File > Update Server).

In the sample image of the group authentication editor shown above, a three-paned display presents the key information required to view and modify groups and member lists. The left pane contains a hierarchical view of authentication groups and users. On the right, the lower pane presents a flat, complete list of existing groups, and the upper pane presents an exhaustive list of individual authenticated users.

Important: Whenever you modify the contents of either authentication data file, then move to an authorization or group editor that uses these member lists, always click the Refresh button to ensure the lists are current.

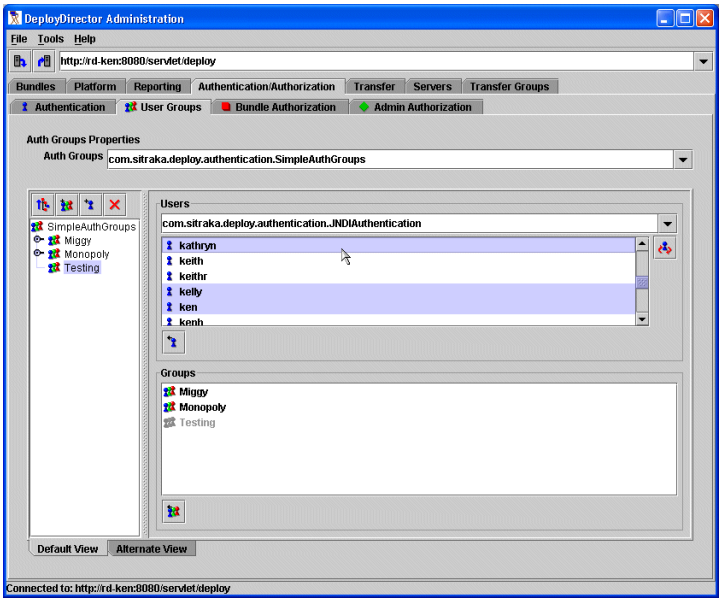
The Alternate View (activated by clicking the appropriately labelled lower tab) presents an easily scanned list of groups and members in tabular format. Although you can enter new groups and users here, the Default View's presentation of group and user information makes it easier to modify authentication groups.

The contents of the list in the top-right pane is dependent on the selected authentication module. The `SimpleAuthentication` module contains the server-based list of authenticated users that you create and manage within DeployDirector. The `JNDIAuthentication` module displays the users found in an organization's NIS directory. When creating or modifying authentication groups, you can use either user list by selecting its module from the User Groups drop-down list.

Modifying the `SimpleAuthentication` data is covered in the next section. The `JNDIAuthentication` user list is not modified with DeployDirector, but you can indicate the location of the directory that will be used. (Please refer to the procedure about authenticating users against an NIS directory in Setting Authentication Properties in Chapter 6.)

Managing Authentication Lists

Creating and modifying the authentication list for individual users simply means working with the user/password data table, which represents `SimpleAuthentication` data. Managing authentication groups (i.e. `SimpleAuthGroups` data) is also straightforward, and can be performed in the authentication groups editor's Default View.



Important: The information displayed in the list is locally cached, thus is as current as the last time you refreshed it. It is important to refresh lists when using this editor.

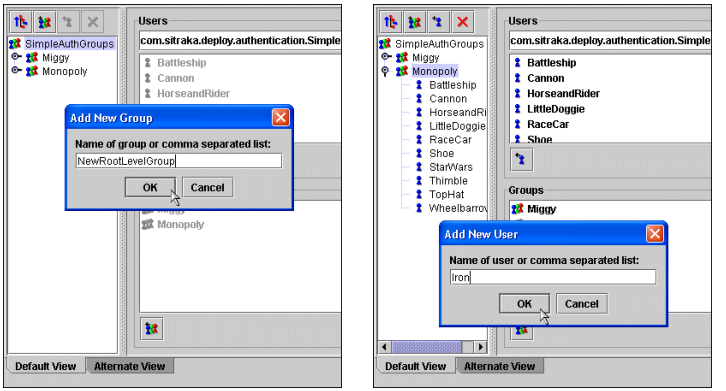
Please refer to An Emphasis On Server Updating and Refreshing, found later in this chapter for more information.

As mentioned in the previous section, the authentication groups and existing members are listed in the left pane in a hierarchical list. The list of groups and members found here are as current as the last time you refreshed the data from the server, as is the case with the list of authenticated individual users in the top-right pane.

When one or more authentication groups are selected in the left pane (multiple items can be selected with key-clicking, for example, Ctrl of Shift-clicking on Windows clients), the contents of the Users list on the right are enabled. Once enabled, you can select one or more user names and add them to the selected authentication group(s). Similarly, you can also select entire groups (listed in the lower-right pane), and add them to selected groups, creating nested authentication groups.

Although it is up to administrators to decide how complex authentication group hierarchies can be, the editor validates created groups to ensure that circular references are not created.

You can also enter new group or user names directly by making selections in the hierarchical authentication groups list. Selecting the list root, or any groups, then clicking the 'add new group' button, allows the creation of a new group named via a pop-up dialog box. Similarly, selecting any group (including nested groups), then clicking the 'add new user' button allows the creation of a new user or users:



Please note that although they are often used together to build authentication lists, the data files associated with individual authenticated users (`SimpleAuthentication`), and authentication groups (`SimpleAuthGroups`) are different, and need to be independently configured. For example, even when you create a new user in an authentication group editor (as shown in the above-right sample screen), it is not automatically added to the authenticated individual users list.

Adding a new user to the authentication data file

1. In the Administration Tool, click the Authentication/Authorization tab, then click the Authentication child tab.
2. In the Authentication drop-down list, select the (`com.sitraka.deploy.authentication.`)`SimpleAuthentication` class.

   The contents of the `SimpleAuthentication` file are displayed. The list of users are individuals that, using their user name and passwords listed, can be allowed to access some or all DeployDirector bundles. Users present on this list can also be individuals who play the role of a DeployDirector administrator.
3. Click New.
4. Enter a name and password for the user.
5. Continue to add more users, then click File > Update Server to update the server data file with these changes.

   The next time you use an authorization editor or group authentication editor that makes use of the `SimpleAuthentication` module, these new users will appear in the Users and Groups list.

Modifying or deleting users in the authentication data file

1. In the Administration Tool, click the Authentication/Authorization tab.
2. If you want to modify the list of individual users, click the Authentication child tab.

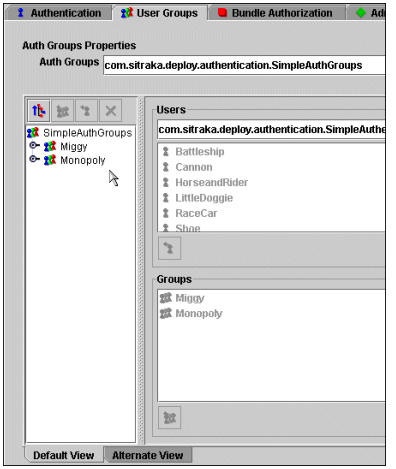
   If you want to modify groups of users, click the User Groups child tab.
3. Click the appropriate field, then modify its contents, or click a field, then Delete.
4. Press the Enter key to commit this deletion.
5. Click File > Update server to ensure these changes are recorded on the server.

   For any bundle that is configured to use either of the "simple" authentication modules, when authorization associations are being configured via the Users and Groups editor, any changes made here will be reflected in the list of users.

Adding new or existing groups to other authentication groups

1. In the Administration Tool, click the Authentication/Authorization tab, then click the User Groups child tab.
2. Click File > Refresh to ensure the data is current.

The authentication groups found in the server data file are listed in the left, and lower-right panes. Expanding the groups in the left pane reveals nested groups, if they exist.



3. In the authentication groups list, select all of the elements to which you want to add a new group.

   To create a top-level group, select the root of the list (labelled as SimpleAuthGroups). Otherwise select any top-level, or subgroups.

4. To create a new group, click the ‘add new group’ button, then enter a name.
   or

   To add one or more existing groups, select them from the Groups list, then click the ‘add selection’ button.

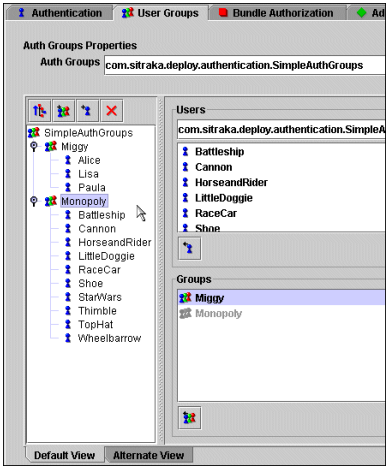
5. If you have created a new group, continue by adding existing members to its membership list (covered in the next procedure). Otherwise, click File > Update Server to update the server data file with these changes.

   Note: The behavior of the Groups list (lower-right pane) depends on what is selected in, or added to, the hierarchical authentication groups list (left pane): any groups added to the authentication groups list subsequently appear in the Groups list. In order to prevent circular nesting, items in the Groups list are disabled when the same groups have been selected in the authentication groups list. They will also be disabled if they are the parent groups of any current selections.

Adding users to authentication groups

1. In the Administration Tool, click the Authentication/Authorization tab, then click the User Groups child tab.
2. Click File > Refresh to ensure the data is current.

   The authentication groups found in the server-side data file are listed in the left pane. Expand the groups whose members you want to view.



3. In the Users drop-down list, select the (`com.sitraka.deploy.authentication.`) `SimpleAuthGroups` class, or the `NISAuthGroups` class, depending which set of authenticated users you want to view and add.
4. In the authentication groups list in the left pane, select the group(s) to which you want to add new users.

Tip: When the working with very long lists of users, type the first few letters of a user name to skip to it in the list. Pausing, then resuming key strokes refocuses on the newly entered letters.

5. In the Users list, select the user or users you want to add to the selected existing authentication group(s).
6. Click the 'add selection' button to add the highlighted users to the selected authentication groups.

7. Whether you have selected the SimpleAuthGroups or NISAuthGroups classes, if at this point you need to add users located in the list you did not choose, select it from the Users drop-down list, and repeated the last two steps.
8. Click File > Update Server to update the server data file with these changes.

The next time you use an authorization editor that makes use of the `SimpleAuthGroups` authentication module, these new groups and users will appear in the Users and Groups list.

Deleting groups from the group authentication list

1. In the Administration Tool, click the Authentication/Authorization tab, then click the User Groups child tab.
2. Click File > Refresh to ensure the data is current.

   The authentication groups found in the server-side data file are listed in the left pane. Expanding the groups in the left pane reveals nested groups, if they exist.
3. Select the group(s) you want to remove from the authentication list.
4.  Click the 'delete' button.

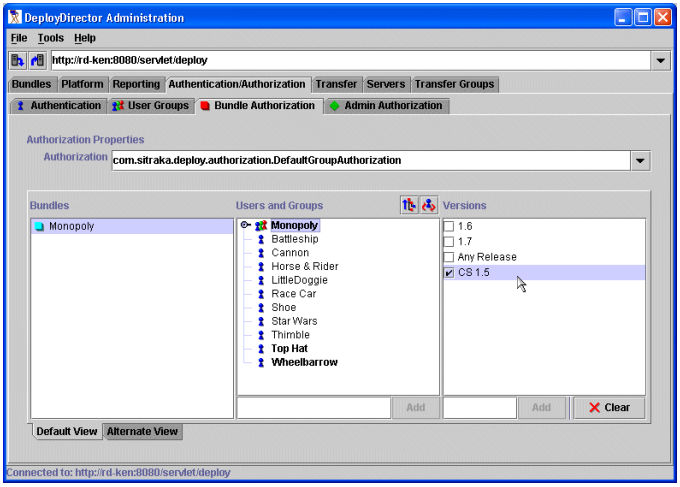
   All instances of the authentication group disappears from both the hierarchical authentication groups list in the left pane, and the Groups list in the lower-right pane.
5. Click File > Update server to ensure these changes are recorded on the server.

   For any bundle that is configured to use this authentication module, when authorization associations are being configured via the Users and Groups editor, any changes made here will be reflected in the list of users.

Viewing End-User Bundle Associations

When you configure bundles (through the Bundles tab or the Bundle Wizard), one set of properties that needs to be configured are the Access properties. Configuring them means declaring which authentication and authorization modules are used, which determines which users and groups have access to them. Furthermore, you may need to declare which users and groups are permitted to access specific bundle versions.

The Bundle Authorization tab (a child tab of the Authentication/Authorization tab) displays authorization associations between bundles, bundle versions, and groups and/or users:



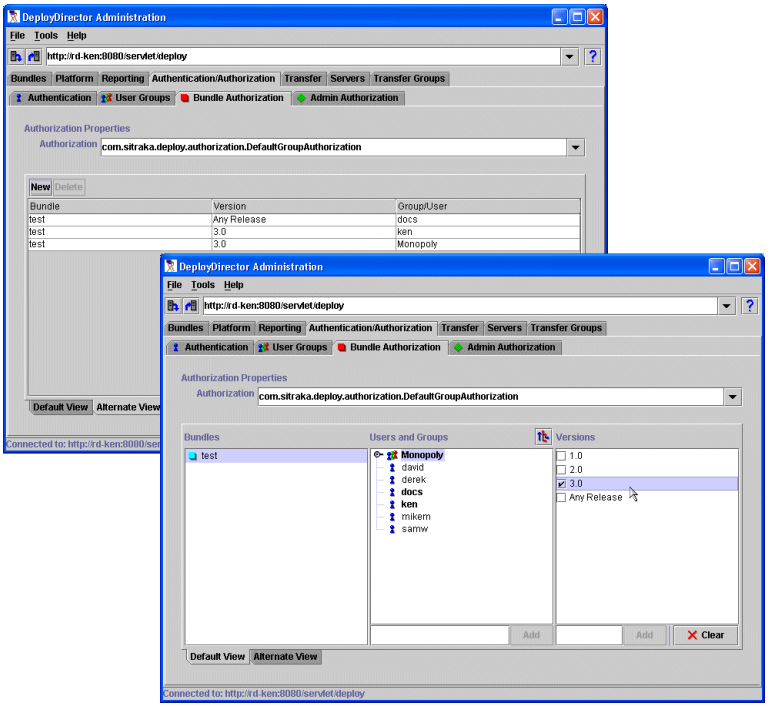
While the Alternate View clearly shows a list of existing users who are authorized to access specific bundle versions, this Default View offers an easy way to set up potentially diverse sets of authorized users.

In each pane, you are shown lists of (from left to right) bundles, users and groups, as well as bundle versions. Setting up authorization associations properly requires an understanding of how these lists are populated, as well as the behavior of these displays.

Default and Alternate Views of End-User Associations

In the section Viewing Authentication Lists on page 161, you learned about the server data files associated with the `SimpleAuthentication` and `SimpleAuthGroups` modules. Using the editors for these two authentication modules (accessed via the Authentication and User Groups child tabs in the Administration Tool), you are able to modify the contents of the data files, thus manage authenticated users.

The editors provided to assist your creation of end-user authorization associations (via the Bundle Authorization child tab) offer this aforementioned “grid” view, as well as the Default View:



Whether using the Default View or the Alternate View, all associations shown and made affect the data file associated with the module selected from the editor's authorization drop-down list (in the above screen images, the `DefaultGroupAuthorization` module is used).

Considering how the views are structured, it is recommended that you use Alternate Views to easily see a list of current associations. Since the features provided in this view are a subset of those found in the Default View, you should use Default Views to actually create and modify associations.

The one task that can be performed exclusively in the Alternate View is the creation of associations that involve elements (i.e. users, bundles or versions) that do not yet exist.

Creating associations for future users or bundles

1. In the Administration Tool, click the Authentication/Authorization tab, then click the Bundle Authorization child tab.

2. In the Authorization drop-down list, select the authorization module whose user-bundle-version associations you want to modify.

The contents of the chosen module and file are displayed with the Default View.

3. Click the Alternate View tab, which is located below the Default View lists.

   This view displays the existing associations through which users are authorized (using the module to which this data file belongs) to use a particular version of a particular bundle.

4. Click New to begin entering a new association.

5. In the User ID or Group/User field, enter the name of an existing or new (nonexistent) user or group.

6. In the Bundle field, enter the name of a vault-based bundle, or a new (nonexistent) bundle.

7. In the Version field, enter the bundle's version name.

8. Continue to add more records, then click File > Update Server to update the server-side data file with these changes.

   Since you have added records for users or bundles that do not yet exist, it is important to ensure these lose ends are eventually tied. If the bundle, user, or version are not created as anticipated, it is recommended that you clean up this nonexistent authorization association by deleting that record. Otherwise you may see broken associations in the authorization editor, as described in condition #3 in Displaying Bundles on page 171.

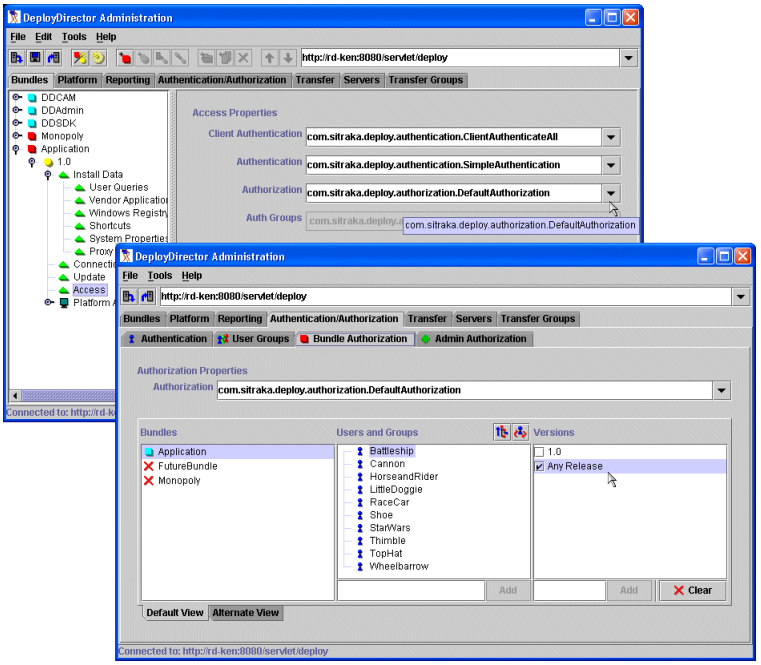
Displaying Bundles

When you click the Bundle Authorization child tab, you are shown lists of bundles, users and versions. The contents of the Bundles list does not necessarily represent every bundle that exists on the server to which the Administration Tool is connected, or the administrator workstation on which the Administration Tool is running. There are three cases in which bundles will be listed.

1. The bundle's latest version's Authorization module matches the module currently selected in the authorization editor.

In the process of configuring a bundle, an administrator selects a particular authorization module that the bundle uses, then may switch over to the authorization editor, then creates associations for users and groups listed for that module.

In the example below, the bundle (named Application), is configured (in the Bundles tab) to use the `DefaultAuthorization` module. This version of the bundle is the only, thus most recent version. When the same authorization module is selected in the authorization editor (shown in the lower screen), the Application bundle appears as the only current bundle in the list. This means none of the latest versions of other bundles on the server, and none of the other uncommitted bundles on the administrator's workstation, are configured to use the `DefaultAuthorization` module.



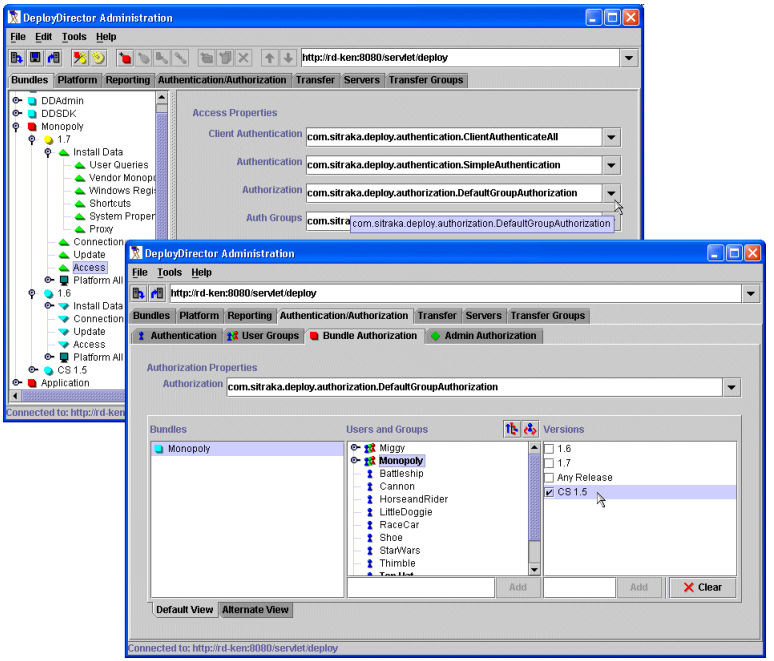
2. A bundle's version has been configured to use an authorization module that matches the module currently selected in the authorization editor, but *it is not the most recent bundle version.*

While it is more common for bundles to always be configured to use the same authentication and authorization modules from version to version, it is entirely possible for these properties to be changed.

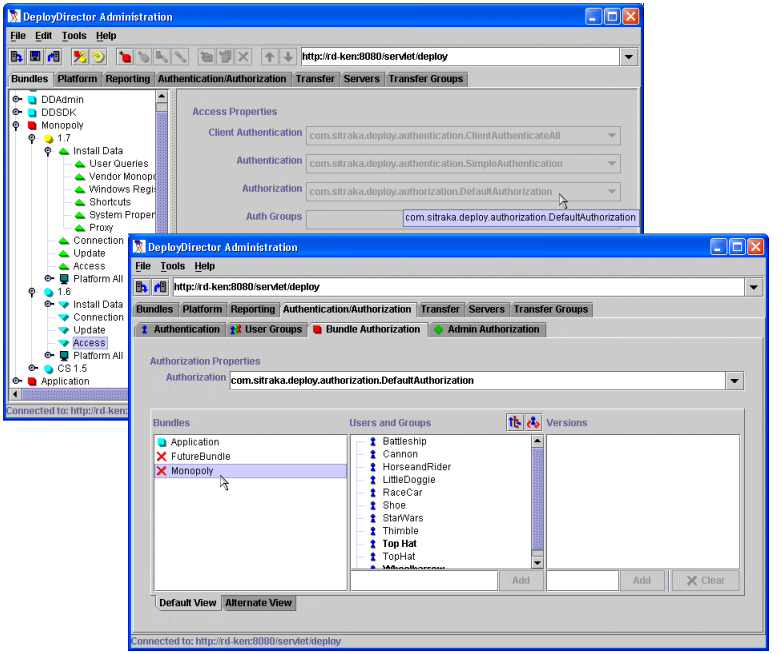
If an administrator, while configuring a new bundle version, selects an authorization module *that differs from that of a previous bundle version*, the bundle will appear in *two* places in the authorization editor.

In the editor, when the selected authorization module matches that of the latest bundle version, the bundle appears as a current one. However, when the authorization module in the editor is changed to match that of an older (i.e. not top-most) version, the bundle will appear as noncurrent (denoted by a red 'X.') This is solely meant to indicate to the administrator that the bundle has mapped authorization associations that do not lead to its latest version.

In the example below, the bundle's most recent version (1.7) is configured to use the `DefaultGroupAuthorization` module. As such, it appears as a current bundle in the authorization editor *when that same authorization module is selected* (see lower screen). The Bundles list shows that no other bundles on the server or the administrator's workstation have versions that use (or used) this authorization module.

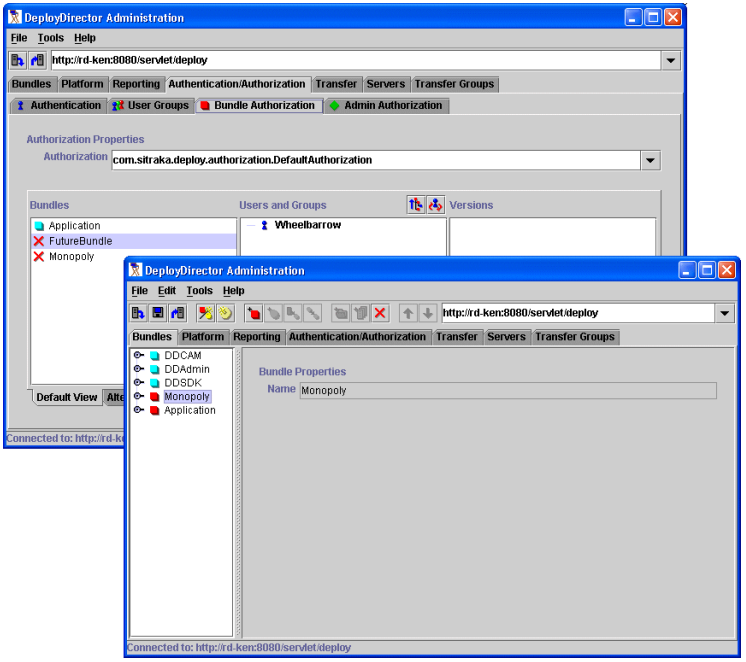


However, looking back at the main Bundles configuration screen (via the Bundles tab), a previous version of the bundle (1.6) used the `DefaultAuthorization` module. After selecting the same module in the authorization editor (as shown in the lower screen), the Bundles list indicates that this bundle is not current. Thus, in this case, the non-current status of the bundle means a previous version was configured to use the authorization module that is currently selected in the editor.

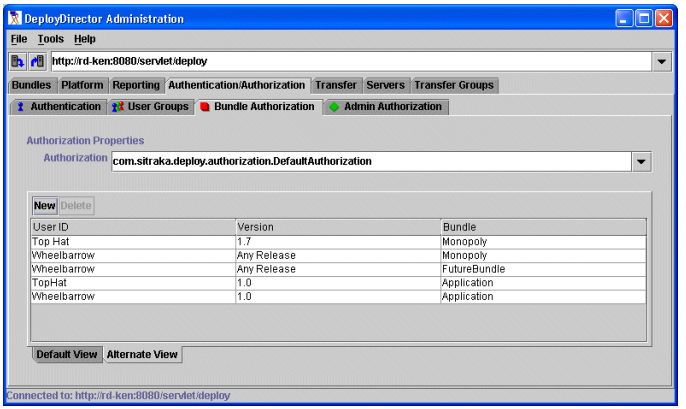


3. An authorization record exists which refers to a bundle that does not yet exist.

In the following example, the `DefaultAuthorization` module has been selected in the editor. Of the three bundles listed, FutureBundle does not appear in the bundle list in the general bundle configuration tree (as shown in the lower screen):



Since it does not appear in the main bundle tree, it is clear that it has not yet been created with the Administration Tool. However, if the `DefaultAuthorization` module's data file is examined using the Alternate View, a reference to FutureBundle is found:



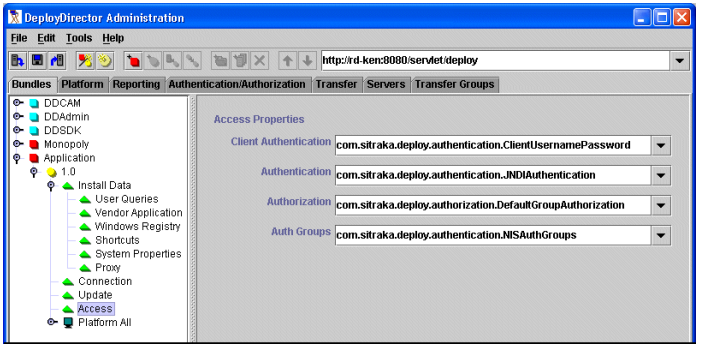
Based on this example, authorization associations that exist in the server-side data file for a given module are always reflected in the editor's Bundles list. This holds true even if the bundle does not exist on the deployment server, and was manually added to the authorization data file. (Perhaps this was the work of a keen administrator, who was preparing for future development projects in the hopes that such foresight would contribute to a bonus that we all know they will never get.)

Displaying Users and Groups

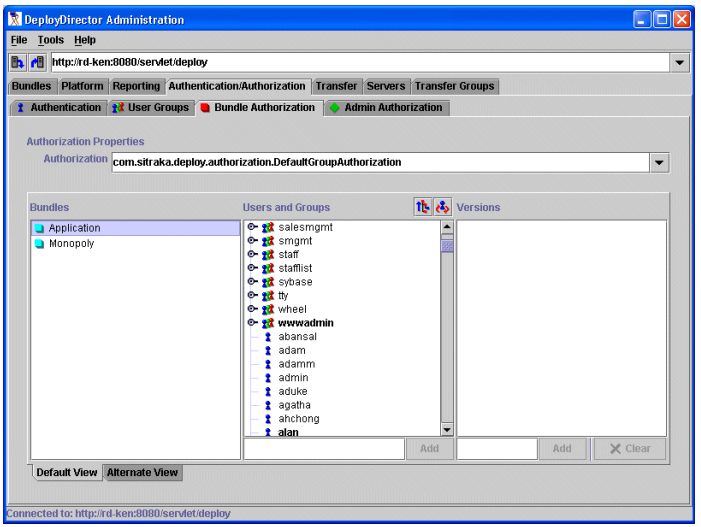
When you click the Bundle Authorization tab to view current, or create new authorization associations, the content of the Users and Groups list is dependent on how bundle properties have been configured. As a rule, DeployDirector will populate the list based on the authentication and authorization modules chosen for a particular bundle, and whether any users exist in those chosen lists.

At the beginning of this chapter, you were introduced to the different default authentication and authorization modules that DeployDirector uses. The following examples demonstrate how the selection of different modules affects the content of the Users and Groups list.

As shown in the main bundle tree, the bundle named Application has been configured to use the `JNDIAuthentication` module to authenticate client-side users, and group authorization lists will be compiled using an organization's NIS directory (`DefaultGroupAuthorization` using the `NISAuthGroups` module).

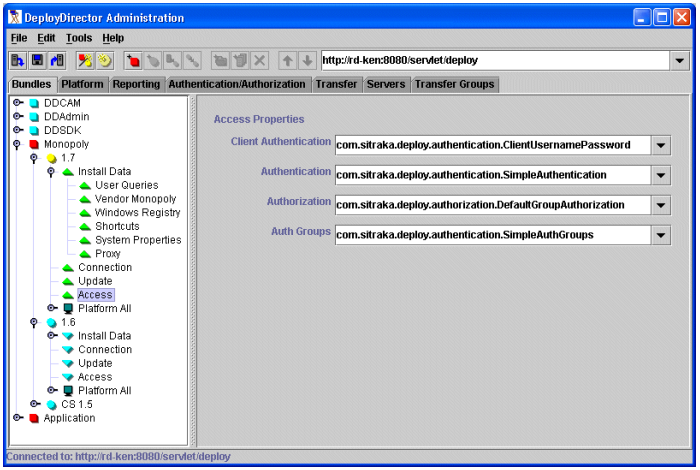


When the Bundle Authorization child tab is clicked, the proper Authorization module must first be selected from the drop-down list. Since the bundle was configured to use the `DefaultGroupAuthorization` module, this is what is selected. Doing so produces a list of bundles in the left pane that are configured to use this module. Selecting the Application bundle populates the Users and Groups list with the appropriate data.

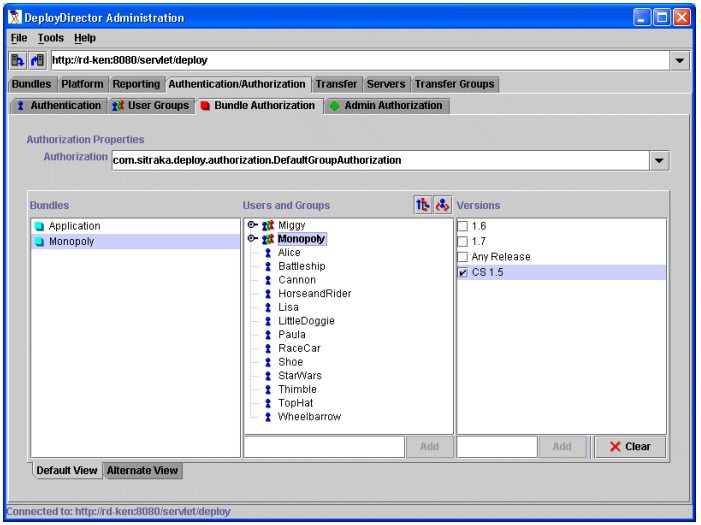


As shown in the above illustration, the list contains all users and work groups in the organization's NIS directories.

As a contrast to the previous example, the next bundle is configured to use DeployDirector's own server-side authentication data files. Here, the bundle has been configured to use the `SimpleAuthentication` module, which authenticates client-side users against its related data file. Additionally, group authorization lists will be compiled based on the contents of the server file associated with the `SimpleAuthGroups` module.



Moving back to the Bundle Authorization child tab, if the same authorization module is selected, and the Monopoly bundle is chosen, the Users and Groups list is populated differently. This time, the contents of this list come from the two data files instead of the NIS directory.



Based on these examples, the authorization editors provide a controlled environment in which you can create authorization associations. Only users relevant to the authentication module chosen during the configuration of the bundle will appear in user lists.

Note: Whenever you modify any authentication lists (in the editors found in the Authentication or User Groups child tabs) that may affect Users and Groups content in the authorization editor, always remember to use the Refresh button to ensure the content is always current.

Manually Adding Names to the Users and Groups List

You can manually add names to the Users and Groups list by entering one the field below the list, then clicking Add.

Once you authorize this user to access a bundle version, this association will be recorded in the related server authorization data file. However, authorized users still need to be authenticated by DeployDirector. It is your responsibility to ensure that this manually added user exists, or is added to the appropriate authentication data file.

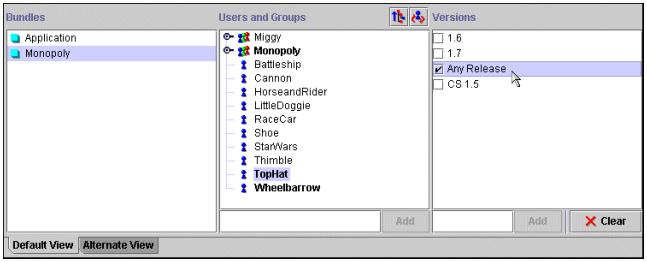
In the most recent example on the previous page, the Users and Groups list in the authorization editor was comprised of authenticated users and groups found in the data files associated with the `SimpleAuthentication` and `SimpleAuthGroups` modules. Adding a new user, and creating an association in the authorization editor would result in a new entry in the `DefaultGroupAuthorization` module's data file, but this new user would not appear in either of the authentication lists/data files automatically.

To avoid potential mix-ups, it is recommended that you first ensure all users are found in the appropriate authentication data files before authorizing them.

Note: If you configure a bundle to use the `WindowsAuthentication` module, the Users and Groups list will not be automatically populated in an authorization editor. In this case, you will have to manually add the Windows user names in the authorization editor. (For each user, enter their user name, then click Add.)

Selecting Bundle Versions

When creating associations in any authorization editor, selecting a bundle, then a user or group will result in the display of at least one bundle version in the right pane. The Versions pane allows you to finish establishing your authorization association by selecting which version of a selected bundle the selected user or group is authorized to use.



The contents of the Versions list is dependent on the number of versions that exist for the selected bundle, and which sets of authorization keywords exist for the authorization module you are using.

By default, DeployDirector's authorization modules include one set of authorization keywords called Any Release, which authorizes a given user to access the latest release. If you are using the DeployDirector SDK to create custom authorization classes, other sets of authorization keywords will also appear in the Versions list.

Managing End-User Bundle Access

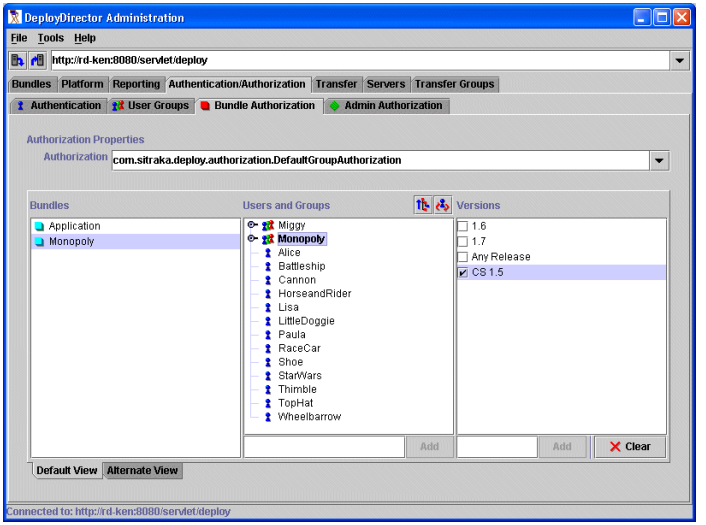
In the previous section, you learned how the authorization editor's Bundles, Users and Groups, and Versions lists are populated. When authorizing users to access a bundle, it is important to remember how bundles and users appear in these lists. From here, the Bundle Authorization editor can be used to easily create associations between users and specific bundle versions.

Authorizing Users or Groups to Access Bundle Versions

In Setting Authorization Properties in Chapter 6, you were shown how to configure bundles to authorize client-side users in different ways. One step in these procedures requires the administrator to review and edit the users or groups that are authorized to use the bundle version that is being configured. Regardless of which authorization module is in focus in the authorization editor, the general process of authorizing users is the same for each one.

When authorizing users by creating associations, a list that is found in the editor is populated based on the selection made before it. The example screen below shows:

- the module selected from the Authorization drop-down list determines which bundles appear (those whose most recent version is configured to use that module),
- the selected bundle's latest version determines which Users and Groups appear (i.e. users that appear are based on the authentication module used by the version),
- the Versions appear when a user or group is selected, and these reflect the versions that exist for that bundle, including uncommitted ones that are currently being configured.



Tip: You can select multiple members in the Users and Groups list by key-clicking (e.g. Ctrl or Shift-clicking on Windows clients).

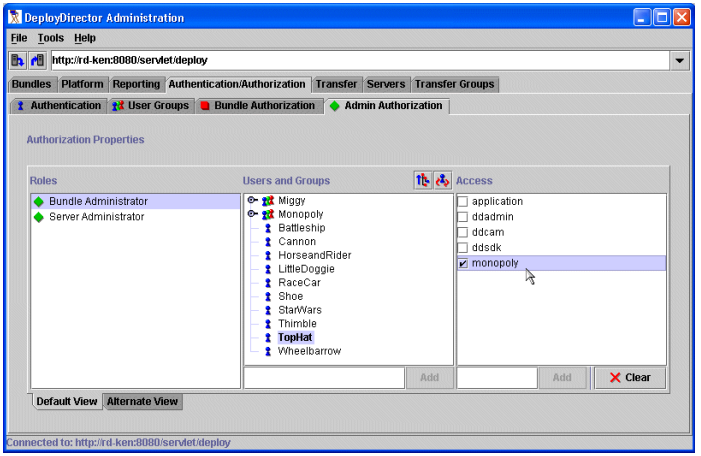
When you select a version, the chosen user or group becomes authorized to access it. This action is confirmed when their name is bold faced.

Finally, always click the Update Server button (also accessed in the menu with File > Update Server).

Viewing Administrator Roles

DeployDirector comes packaged with a default “super administrator” role (initially discussed in Chapter 1, Installation and Setup). Your organization may also require the presence of administrators that have their own domains, in which specific bundles require their attention.

The Admin Authorization tab (a child tab of the Authentication/Authorization tab) displays authorization associations between users (acting as administrators) and bundles, and which users have server access:



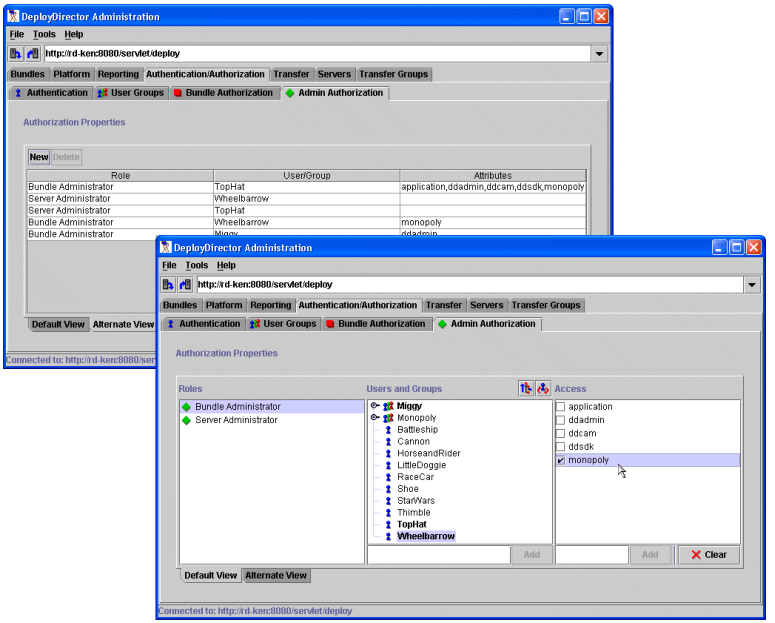
As with the authorization editors, the Alternate View shows existing users who are authorized to administer bundles or servers, and the Default View offers an easy way to actually set up authorized users.

Similar to the authorization editors, in each pane, you are shown lists (from left to right) of Roles, Users and Groups, and Access. Using these lists to grant administrator status requires an understanding of how they are populated.

Default and Alternate Views of Administrator Associations

In the section Default and Alternate Views of End-User Associations on page 169, you learned about the two views provided by authorization editors: Default Views and Alternate Views.

The editors provided for authorization users to administrate bundles and servers offer the same types of views:



It is recommended that you use the Alternate Views to view lists of current associations (e.g. in the top screen, above). Since the features provided in this view are a subset of those found in the Default View, you should use Default Views to actually create and modify associations.

The one task that can be performed exclusively in the Alternate View is the creation of associations that involve elements a user or group that does not yet exist.

Authorizing future (nonexistent) administrators

1. In the Administration Tool, click the Authentication/Authorization tab, then click the Admin Authorization child tab.
2. Click the Alternate View tab, which is located below the Default View lists.

   This view displays the existing associations through which users are assigned roles, and are authorized to manage servers, or specific bundles.
3. Click New to begin entering a new association.
4. In the Role field, select whether this new user is a Server Administrator, or Bundle Administrator.
5. In the User/Group field, enter the authenticated name of the new administrator body.
6. In the Attributes field, if the new administrator will manage bundles, enter the name of the vault-based bundle. If the user is a Server Administrator, leave the field empty.
7. Continue to add more records, then click File > Update Server to update the server-side data file with these changes.

Since you have added records for users that do not yet exist, it is important to ensure these lose ends are eventually tied. If the user is not created as anticipated, then it is recommended that you clean up this nonexistent authorization association by deleting that record.

Displaying Users and Groups

When you click the Admin Authorization tab to view current, or create new administrator authorization associations, the content of the Users and Groups list is dependent on how the server's `cluster.properties` file has been configured. As a rule, DeployDirector will populate the Users and Groups list based on the authentication and authorization modules set in the `cluster.properties` file, and whether any users exist in those chosen lists.

The two cluster properties referenced by the Administration Tool to populate Users and Groups lists are:

`deploy.admin.authentication`
and
`deploy.admin.authgroups`

Tip: Please see End-User and Administrator Authentication Lists on page 160 for more information about these standard authentication modules / classes.

At the beginning of this chapter, you were introduced to the different default authentication and group authorization modules that are used by DeployDirector. One of these modules (or a custom class you create with the DeployDirector SDK) is referenced. By default, these cluster properties are respectively set to:

`com.sitraka.deploy.authentication.SimpleAuthentication`
and
`com.sitraka.deploy.authentication.SimpleAuthGroups`

These modules can be changed so that other authentication sources are used to populate the users list. To do this, you need to modify your `clusters.properties` file.

In the following procedure, the source of the list of authenticated users will be changed. Instead of retrieving a list of users from the `SimpleAuthentication` and `SimpleAuthGroups` data files, the editor will reference an organization's NIS directory.

Despite the fact that this user list source can be changed, it is likely that the number of administrators in your organization will be small enough, that the maintenance of the DeployDirector-based data files will introduce negligible overhead.

Changing the administrator Users and Groups source

1. Shut down the DeployDirector server whose administrator roles you wish to reconfigure.
2. In a text editor, open the `cluster.properties` file, which is found in the `<installpath>/deploydirector/` directory.
3. Locate the `deploy.admin.authentication` property, and replace the currently referenced class with another DeployDirector authentication class, or your own custom class, ensuring you enter its full package.

   For this example, the new class for user authentication could be `com.sitraka.deploy.authentication.JNDIAuthentication`, which will authenticate users against their JNDI login information.

4. Locate the `deploy.admin.authgroups` property, and replace the currently referenced class with another DeployDirector class, or your own custom class, ensuring you enter its full package.

   For this example, the new class for group authentication will be `com.sitraka.deploy.authentication.NISAuthGroups`, which will authenticate all user groups from the NIS directory.

5. Save the changes you have made to the `cluster.properties` file.

6. Restart the server.

7. Run the Administration Tool.

8. Click the Authentication/Authorization tab, then click the Admin Authorization tab.

   Clicking either the Bundle Administrator or Server Administrator roles will result in the populating of the Users and Groups list. The contents of this list will reflect the changes made to the `cluster.properties` file, since the new authentication classes used will produce different lists of people.

   In this example, since the `SimpleAuthentication` and `SimpleAuthGroups` modules were replaced by JNDI/NIS related modules, the contents of the Users and Groups list will be based on users found on your company's directory, as opposed to the contents of the simple server data files.

Managing Administrator Access

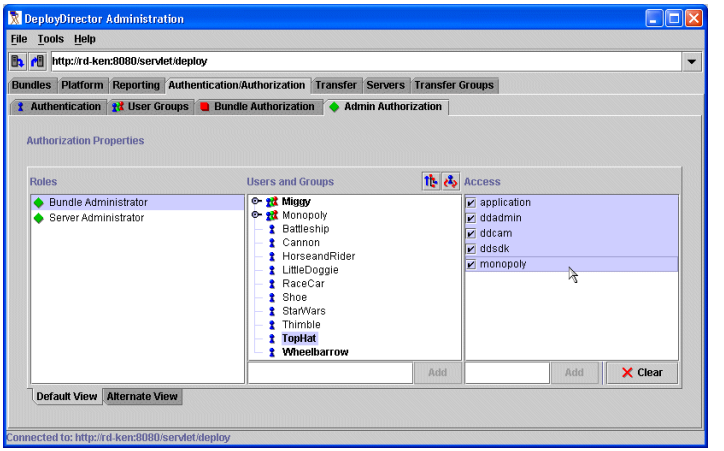
In the previous section, you learned how certain groups of authenticated users (based on a source you have defined) appear in the Admin Authorization editor's Users and Groups list. To manage administrator roles and access, use the Admin Authorization editor to define roles for these users, which includes bundle administrators, and server administrators. This is performed by creating associations between users and roles.

Defining Bundle Administrators

Bundle Administrators can be defined to manage any number of bundles. They may have complete access to all an organization's bundles, or if the organization is quite large or has rigidly separated departments, bundle administrators may also have access to a very specific subset of bundles.

When a member has been selected in the Users and Groups list, all bundles found on the server appear in the Access list in the right pane. In this list, bundles that are selected fall under the jurisdiction of the user.

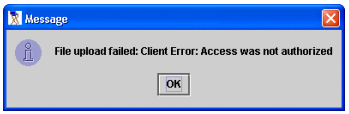
Tip: Multiple bundles can be selected from the Access list by key-clicking (e.g. Ctrl and Shift-clicking on Windows clients).



Once this association has been updated on the server, that user will be able to log in the Administration Tool and make use of their limited power.

When it comes to bundle configuration, the actions that Bundle Administrators can perform, and the privileges they have, are identical to those of top-level administrators. However, Bundle Administrators will not have the ability to modify any server-related settings, which include JRE management, server-side authentication, authorization modifications, or transfer group management. Additionally, they will have limited access to the Remote Administrator.

Additionally, while they will be able to see all the bundles that exist in the vault, they will not be permitted to update the server with any changes they make to a bundle to which they have not been granted administrator access.



Assigning users to administrate specific bundles

1. In the Administration Tool, click the Authentication/Authorization tab, then click the Admin Authorization child tab.
2. In the Roles list, click Bundle Administrator.

   Selecting a Role populates the Users and Groups list, whose contents are dependent on user/group source files declared in the main `cluster.properties` file.
3. From the Users and Groups list select a member (whether it is an individual user or a group).

   Selecting a member from Users and Groups populates the Access list, which consists of all the bundles that are found in the server vault.
4. Select all the bundles to which this user is permitted to administrate. Use key-clicking (e.g. Ctrl-clicking and Shift-clicking on Windows clients) to make multiple non-contiguous selections.

   When a selection is made, the user name is bold-faced to indicate that an association exists between a user and one or more bundles.
5. Continue to promote more users the Bundle Administrator role, then click File > Update Server to commit these changes.

Defining Server Administrators

Server Administrator privileges are identical to those of the default administrator profile that comes with DeployDirector. While you can create other custom roles, DeployDirector offers one default Server administrator role. Those who are given this role will be able to access and modify all aspects of the DeployDirector system, which includes functions available in both the Administration Tool and the Remote Administrator.

The process of assigning users the Server Administrator role is very similar to that used for declaring Bundle Administrators.

Assigning users to be server administrators

1. In the Administration Tool, click the Authentication/Authorization tab, then click the Admin Authorization child tab.
2. In the Roles list, click Server Administrator.

   Selecting a Role populates the Users and Groups list, whose contents are dependent on user/group source files declared in the main `cluster.properties` file.
3. From the Users and Groups list select a member (whether it is an individual user or a group).

   Selecting a member from Users and Groups populates the Access list, which by default contains one choice: complete server access.
4. Select the Access option to assign the chosen user or group the Server Administrator Role.

   When a selection is made, the user name is bold-faced to indicate that an association exists between a user and one or more bundles.
5. Continue to promote more users the Server Administrator role, then click File > Update Server to commit these changes.

An Emphasis On Server Updating and Refreshing

Throughout this chapter, as well as in End-User Authentication and Authorization in Chapter 6, you were reminded to use the server Refresh button (or, File > Refresh in the menu) to retrieve the latest user information from server data files, and to always finish any procedure by using the Update Server button (or, File > Update Server in the menu) to make sure your changes are sent to the server.

Most sections in this chapter refer to the various server data files, accessed and modified by the authentication or authorization editors in the Administration Tool.

In a single-administrator environment, it is good practice to refresh the user data before modifying and uploading changes to the server; in a multi-administrator environment, it is *vital* to do so.

The data files viewed in the Administration Tool editors are *locally cached versions* of the server data files. Only when you use the (server) Refresh and Update Server commands can you be assured that the data you see, and the data you have modified, will truly match that which exists on the server.

In multi-administrator environments, there exists the possibility that two administrators may view and modify the same server data file from separate workstations. Whether or not administrative precautions are made to avoid situations like this one, whenever any administrator is managing data in any of the authentication or authorization editors, it is recommended that they:

- Click the (server) Refresh button (or use File > Refresh) to ensure current server data is being used, before performing any actions.
- Make changes in the authentication or authorization editor.
- Commit their changes to the server by clicking the Update Server button (or use File > Update Server).

Customizing the Default Module and Editor Classes

As mentioned earlier in this chapter, client and server-side authentication, and authorization can be customized using the DeployDirector SDK. While the default classes cover a variety of authentication and authorization possibilities, there may be specific issues within your deployment process that require modifications to these classes, or the creation of new ones.

By customizing the authentication classes, you can determine:

- what kind of authentication editor is used in the bundle installer,
- what type of authentication information is required from the user,
- where user authentication profiles are stored,
- in what form user profiles are stored.

By customizing the authorization classes, you can determine:

- how authorization information is stored,
- where user authorization profiles are stored.

It is recommended that you extend the existing DeployDirector classes, as they provide features that can act as the foundation for enhancements.

Chapter 10
Viewing and Managing Logs

Keeping track of all client and server activity on your organization's or department's deployment network is facilitated by DeployDirector's reporting functionality. During the deployment process, servers automatically log information that can help system administrators keep track of and analyze deployment activity. By default, generated logs are stored as server-side files, but can also be written to a JDBC-compliant database.

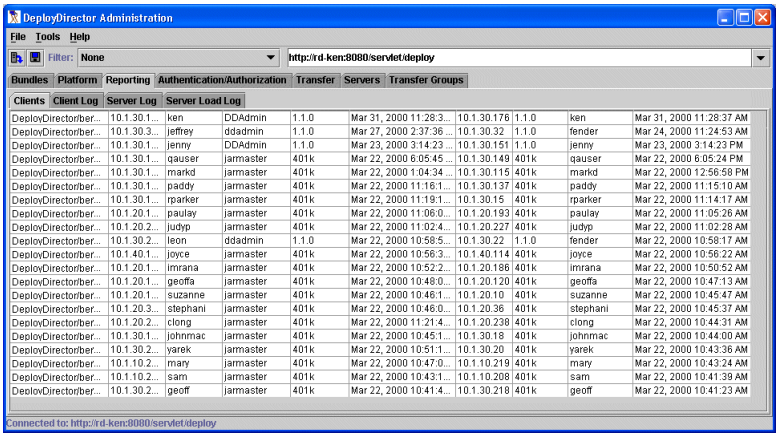
Overview of DeployDirector Logs

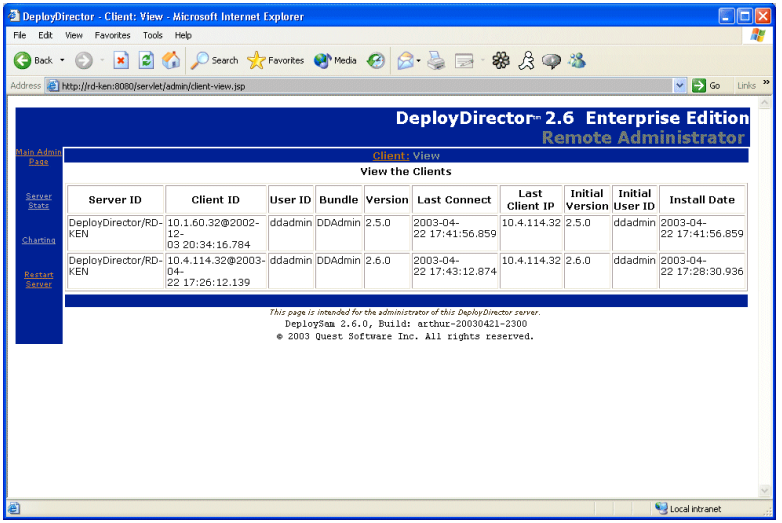
DeployDirector generates four types of logs: the Clients database, the Client Log, the Server Log and the Server Load Log. These can easily be viewed by clicking the Reporting tab in the Administration Tool, or by clicking the appropriate link in the Remote Administrator. (It is recommended that you use the Administration Tool to view log reports, as the information can be sorted by column.)

Clients Database

Part database, part log, all cop. The Clients database provides an easy way of keeping track of all client-side end users who have requested the installation of bundles. The log provides a summary of client-side users, the bundle versions they have downloaded and installed, as well as the details surrounding the installation of those bundles.

To view the Clients database, click the Clients tab in the Administration Tool, or navigate to the Client: View page in the Remote Administrator.





ServerID: The server from which the client-side user downloaded the bundle. (The value shown is either the machine's IP address or its name.)

Client ID: The ID associated with the client machine's CAM.

User ID: The client machine user's name.

Bundle Name: The bundle downloaded by the client-side user.

Bundle Version: The bundle version downloaded by the client-side user.

Last Connection: The last time the client-side user connected to the server through an application bundle.

Last Client IP: The most recent IP address from which the client-side user connected to the server.

Initial Version: The first version of the particular bundle the client-side user downloaded and installed.

Initial User ID: The user ID used to authenticate and authorize the client-side user.

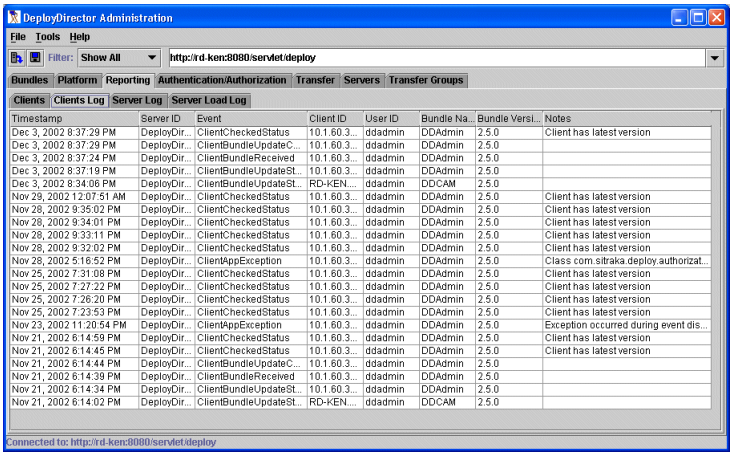
Install Date: The date and time the bundle was installed on the client machine.

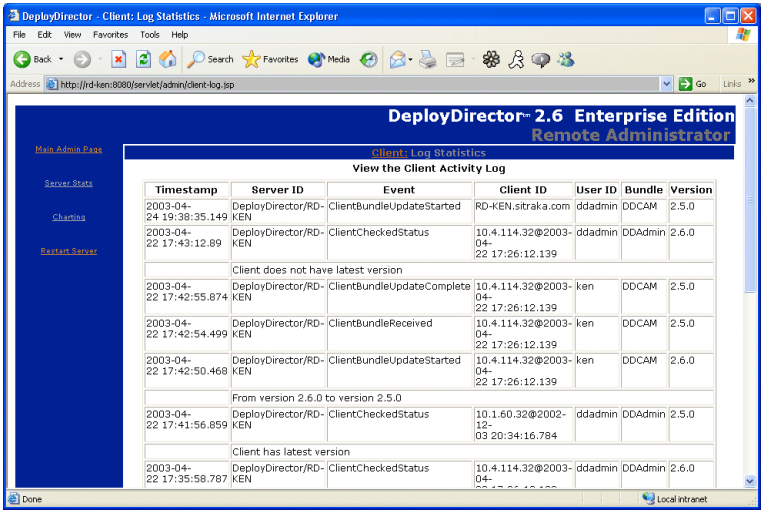
Client Log

The Client Log is used to monitor client-side events. It summarizes all events and errors that occur during client-server interaction and client machine activity. Events and errors can occur during bundle deployment, installation and execution.

By default, bundles are configured so that client-side exceptions are written to the Client Log (as well as printed to the console and a user dialog box). This setting can be turned off, and the information can be written to a separate file. (Please refer to Client-Side Exception Handling and Output in Chapter 6, Configuring Bundle Runtime Properties.)

To view the Client Log, click the Client Log tab in the Administration Tool, or navigate to the Client: Log Statistics page in the Remote Administrator.





Timestamp: The date and time on which the logged event or message was generated.

Server ID: The server to which the client-side user was connected when the logged event or message was generated.

Event: The event that prompted the generation of the log entry.

Client ID: The ID associated with the client machine's CAM.

User ID: The name of the client machine user linked to the event or message.

Bundle Name: The bundle downloaded to, or running on, the client machine.

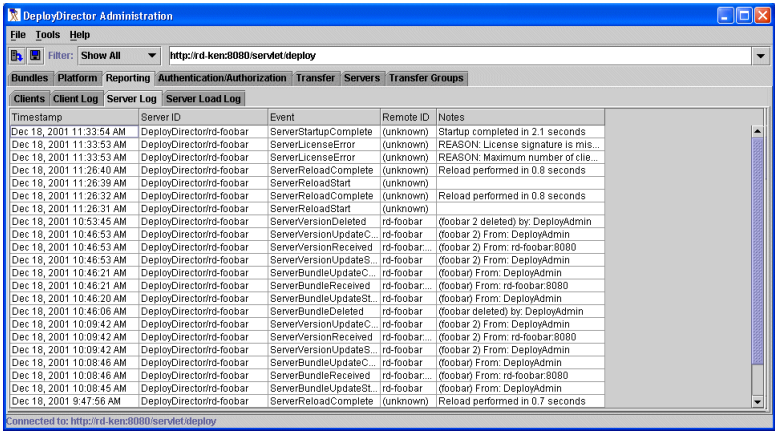
Bundle Version: The bundle version downloaded to, or running on, the client machine.

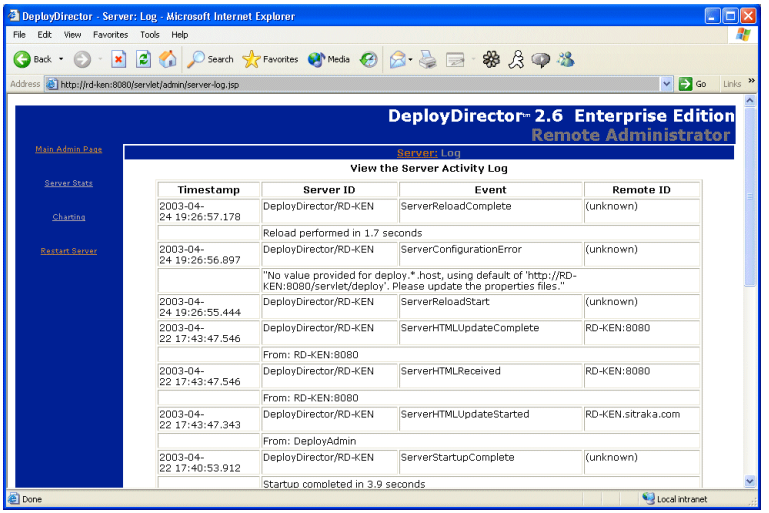
Notes: Any generated messages by the Java Virtual Machine or the Web browser.

Server Log

The Server Log is used to monitor all server-side activity. It summarizes all events and errors that occur during server activity and server-server interaction. Events and errors can occur during bundle and log replication, cluster logging and inter-server communication sessions.

To view the Server Log, click the Server Log tab in the Administration Tool, or navigate to the Server: Log page in the Remote Administrator.





Timestamp: The date and time on which the logged event or message was generated.

Server ID: The server for which the logged event or message was generated. (The value shown is either the machine's IP address or its name).

Event: The event that prompted the generation of the log entry.

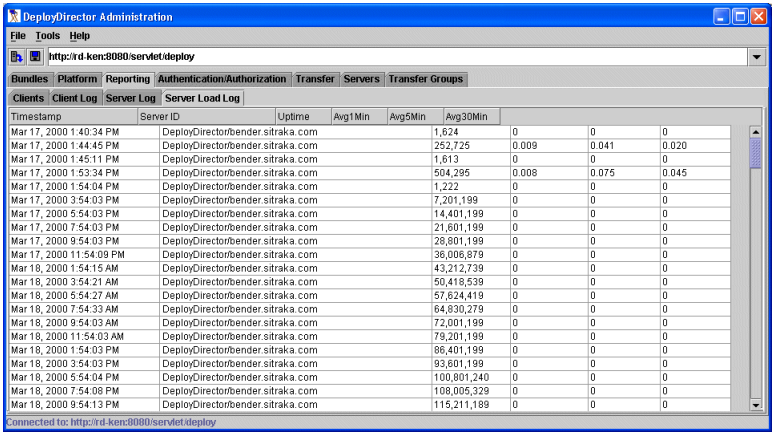
Remote ID: The machine or user with which the server was communicating when the event occurred. (This machine could be another server or a client machine.)

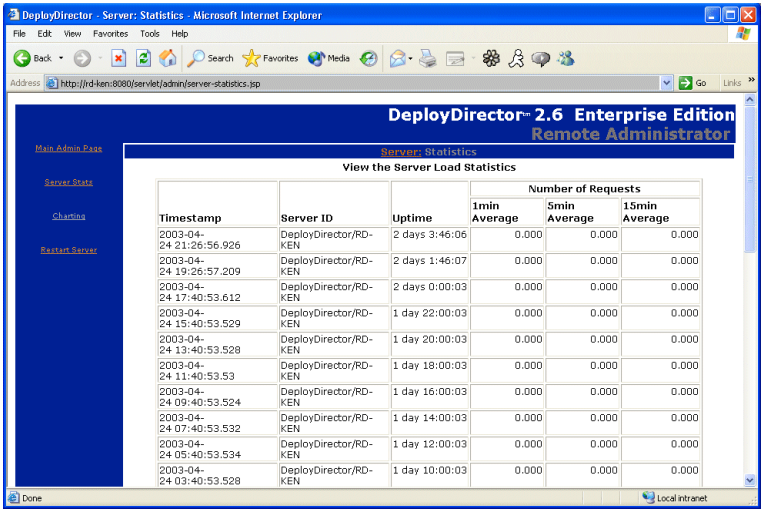
Notes: A synopsis of the details surrounding the logged event.

Server Load Log

The Server Load Log offers a summary of the workload being handled by the servers on the deployment network at a particular moment in time. This log is helpful in determining whether the number of servers on the network is sufficient to handle the number of client-side requests.

To view the Server Load Log, click the Server Load Log tab in the Administration Tool, or navigate to the Server: Statistics page in the Remote Administrator.





Timestamp: The date and time on which the logged event or message was generated.

Server ID: The server to which the client-side user was connected when the logged event or message was generated. (The value shown is either the machine's IP address or its name).

Uptime: The number of milliseconds that have passed since the server has been active.

Avg1Min: The average number of client and server requests processed by the server since it has been active. (The value indicates number of requests per minute.)

Avg5Min: The average number of client and server requests processed by the server since it has been active. (The value indicates number of requests per five minute interval.)

Avg30Min: The average number of client and server requests processed by the server since it has been active. (The value indicates number of requests per thirty minute interval.)

Configuring Log Generation and Storage

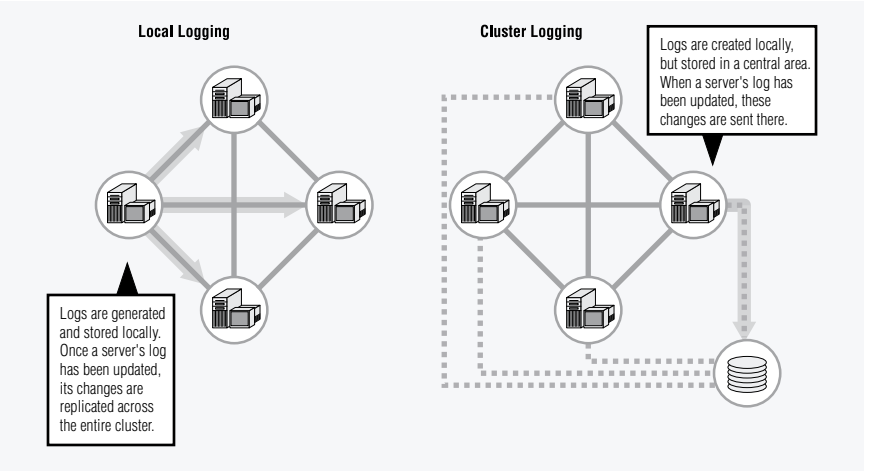
You can configure how logs are compiled in the Remote Administrator. Cluster logging properties are set at the Server: Cluster Configuration: Status Logging page, and individual server logging properties are set at the Server: Server Configuration: Status Logging page. While logging properties can be set at both the cluster and server levels, it is recommended that logging behavior is first set at the cluster level, followed by server-level tweaking.

Since all deployment servers are generating their own respective logs, it is ideal that these logs are all combined and stored in the same location. DeployDirector allows the storing of logs either locally as a flat data file, or centrally in a JDBC-compliant database. This structure ensures access to all server logs, regardless of the server to which the Administration Tool is connected. Setting up this structure is achieved by defining logging properties strictly at the cluster level.

Later in this section, you will be shown which logging properties can be set at the server level to override cluster-level settings. For background information on setting cluster and server properties in general, please refer to Servers and Server Clusters in Chapter 3, Managing Servers and Clusters.

Configuring Logging Methods

DeployDirector supports two types of logging methods: local logging and cluster logging. Enabling local logging results in all deployment servers storing logs locally, with replication to all other servers on the deployment network. (The log replication process is similar to the one used during bundle replication.) Enabling cluster logging results in all server logs being stored directly in an external database.



When an organization uses more than one deployment server, it is important that logging properties are set correctly, ensuring that all logs are properly combined or replicated. Updating the data source (whether a local aggregated log, or a central database) can be performed at set intervals, instead of every time a new log entry is generated. This batching of log entries frees up system resources.

If local logging is enabled, setting aggregate logging properties (i.e. Aggregate Start Date, and Aggregate Interval) ensures that locally logged information is combined with the other server logs. This is carried out when servers contact each other at specified times and upload log entries generated since the last update.

If cluster logging is enabled, setting the JDBC logging properties ensures proper access to the external database, where all server logs are merged.

Setting local logging to flat file

1. In the Remote Administrator, navigate to the Server: Cluster Configuration: Status Logging page.

   No matter which server the Remote Administrator is connected to, any saved changes made at the cluster level are replicated to all other servers that are part of the cluster.

2. Ensure the Local Log check box is enabled (which automatically disables the Cluster Log check box).

3. Ensure the Log to File check box is enabled (which automatically disables the Log to Database check box).

4. In the Log Location text field, verify or enter a new path to which all servers in the cluster will store logs.

   The root of the default value in this field, `$(VAULTDIR)`, is the location of the DeployDirector installation on the server. Changing this location to another part of the server requires a full, hard coded directory path.

5. In the Aggregate Start Date text field, enter the value that indicates the date and time on which log aggregation will begin. (Please refer to the section entitled Administration Tool Date and Time Entry Formats in Chapter 2, Introduction of the Administrator's Guide for a list of valid values.)

6. In the Aggregate Interval text field, enter the value that indicates how often the server will notify other servers in the cluster that it has new log entries to share.

   Now that these two properties have been defined, the deployment servers will begin sharing log entries with each other at the specified time intervals.

7. If desired, continue to set other logging properties, which include those that affect logging limits, and log writing frequency.

8. Click Set Configuration to save changes, then restart the server.

Setting cluster logging to a JDBC database

1. Start the database server.

2. Ensure that a JDBC driver for that database is available to DeployDirector (e.g. if are you using Tomcat, drop the JAR into the `/lib` directory of your DeployDirector installation).

3. In the Remote Administrator, go to the Server: Cluster Configuration: Status Logging page to display cluster level logging properties.

   No matter which server the Remote Administrator is connected to, any saved changes made at the cluster level are replicated to all other servers that are part of the cluster.

4. Ensure the Cluster Log check box is enabled.

   Setting this property enables cluster logging. Since cluster logging results in log writing to an external database, you need to set some pertinent JDBC settings.

5. Ensure the Log to Database check box is enabled (which automatically disables the Log to File check box).

6. In the JDBC Driver text field, enter the name of the JDBC driver.

   Setting this property may not be required if a JDBC driver has already been loaded as part of the servlet or Web server environment.

7. In the JDBC URL text field, enter the URL used to connect to the database.

   The name of the database in which DeployDirector will create data should be included in the URL.

   For example, entering `jdbc:sybase:Tds:YourMachine:4000/YourDB` instructs DeployDirector to connect to a Sybase database, connect to a machine named `YourMachine` on port 4000, and to create all required tables in the `YourDB` database.

8. In the JDBC User and JDBC Password text fields, enter the login user name and password for a user with table creation and update permissions for the database.

9. If desired, continue to set other logging properties, which include those that affect logging limits, and log writing frequency.

10. Click Set Configuration to save changes, then restart the server.

Once cluster logging has been enabled, JDBC information has been set and the changes have been committed to the server, log entries are written to the central database on the fly as they are generated.

Configuring Logging Limits

When local logging is enabled (i.e. the Local Log check box is enabled on the Server: Cluster Configuration: Status Logging page), logs are aggregated and stored on each deployment server. By default, logs are held for a maximum of 30 days before being deleted, as long as the defined minimum of 500 log entries exists. Additionally, servers store a maximum of 1000 logs, regardless of how old they are, to facilitate log management during heavy deployment periods.

These default properties can be set at the cluster level to match your organization's needs. (For example, you may want to increase the maximum number of logs and the length of time they are kept if your servers deploy bundles frequently.)

Setting cluster level log limit properties

1. In the Remote Administrator, navigate to the Server: Cluster Configuration: Status Logging page and locate the Log Limits property text fields.

   No matter which server the Remote Administrator is connected to, any saved changes made at the cluster level are replicated to all other servers that are part of the cluster.

2. In the Minimum Size text field, enter the minimum number of log entries required before they are deleted.

3. In the Maximum Size text field, enter the maximum number of log entries that can be stored on a server.

4. In the Maximum Age text field, enter a value that represents the amount of time a log entry is kept before being deleted. (Please refer to Administration Tool Date and Time Entry Formats in Chapter 2, Introduction for a list of valid values.)

5. Click Set Configuration to save changes, then restart the server.

   Once changes have been committed, the settings made will be replicated to all other servers in the cluster.

Configuring Log Writing Frequency

Whether local or cluster logging is used, all events and errors that are part of the Client Log or Server Log are written to them. Log entries written to a JDBC database are done so immediately, while those destined for a local flat file are batched and written at defined intervals.

In the case of the Clients database and the Server Load Log, the frequency of log writing depends on the defined frequency of their respective "snapshots." This can be defined by configuring the Clients Logging and Load Logging, respectively, using the Remote Administrator. More frequent snapshots offer more information; however, to reduce overhead, you can take infrequent snapshots, or disable either log altogether.

Setting the frequency of snapshots for the Clients database

1. In the Remote Administrator, navigate to the Sever: Cluster Configuration: Status Logging page and locate the Clients Logging property text fields.

   No matter which server the Remote Administrator is connected to, any saved changes made at the cluster level are replicated to all other servers that are part of the cluster.

2. In the Update Start Date text field, enter the value that indicates the date and time on which the Clients database will be updated for the first time. (Please refer to Administration Tool Date and Time Entry Formats in Chapter 2, Introduction for a list of valid values.)

3. In the Update Interval text field, enter the value that indicates how often the Clients entries are written to the database.

   Entering a value of `-1` disables any writing to the Clients database.

4. Click Set Configuration to save changes, then restart the server.

Setting the frequency of snapshots for the Server Load Log

1. In the Remote Administrator, navigate to the Server: Cluster Configuration: Status Logging page and locate the Load Logging property text field.

   No matter which server the Remote Administrator is connected to, any saved changes made at the cluster level are replicated to all other servers that are part of the cluster.

2. In the Log Frequency text field, enter the value that indicates how often a snapshot of the server's current load is taken. (Please refer to Administration Tool Date and Time Entry Formats in Chapter 2, Introduction for a list of valid values.)

3. Click Set Configuration to save changes, then restart the server.

Overriding Cluster Logging Settings for a Server

Configuring logging properties is normally done at the cluster level. However, you can also set certain properties for individual servers, which effectively overrides the equivalent cluster level setting.

To set logging properties for a particular server, connect to it with the Remote Administrator, and configure properties on the Server: Server Configuration: Status Logging page, exactly as you would cluster properties.

At the server level, you can set and override cluster-level settings for:

- local log file locations (for local logging configurations)
- server aggregation intervals (for local logging configurations)
- Server Load Log and Clients database writing frequency
- logging limit properties.

Typically, overriding cluster-level settings with server-specific ones occurs when differing server hardware specifications (e.g. free hard drive space) need to be balanced out.

Overriding where the logging file is stored

1. In the Remote Administrator, navigate to the Server: Server Configuration: Status Logging page.

2. In the Log Location text field, enter the path on which logs are to be stored for this particular server.

   The path set at the cluster level is overridden. When you restart the server, logs will be stored at the specified path.

3. Click Set Configuration to save changes, then restart the server.

Overriding how often a server aggregates logs

1. In the Remote Administrator, navigate to the Server: Server Configuration: Status Logging page.
2. In the Aggregate Start Date text field, enter the value that indicates the date and time on which log aggregation will begin for that particular server. (Please refer to Administration Tool Date and Time Entry Formats in Chapter 2, Introduction for a list of valid values.)
3. In the Aggregate Interval text field, enter the value that indicates how often the server will notify other servers in the cluster that it has new log entries to share.

   Now that these two properties have been defined, the deployment servers will begin sharing log entries with each other at the specified time intervals.
4. Click Set Configuration to save changes, then restart the server.

Overriding the frequency of Clients database snapshots

1. In the Remote Administrator, navigate to the Server: Server Configuration: Status Logging page and locate the Clients Logging property text fields.
2. In the Update Start Date text field, enter the value that indicates the date and time on which the Clients database will be updated for the first time. (Please refer to Administration Tool Date and Time Entry Formats in Chapter 2, Introduction for a list of valid values.)
3. In the Update Interval text field, enter the value that indicates how often the Clients entries are written to the database.

   Entering a value of -1 disables any writing to the Clients database.
4. Click Set Configuration to save changes, then restart the server.

Overriding when server load logs are sent

1. In the Remote Administrator, navigate to the Server: Server Configuration: Status Logging page, and locate the Load Logging property text field.
2. In the Log Frequency text field, enter the value that indicates how often a snapshot of the server's current load is taken. (Please refer to Administration Tool Date and Time Entry Formats in Chapter 2, Introduction for a list of valid values.)
3. Click Set Configuration to save changes, then restart the server.

Overriding logging limit properties

1. In the Remote Administrator, navigate to the Server: Server Configuration: Status Logging page, and locate the Log Limits property text fields.

2. In the Minimum Size text field, enter the minimum number of log entries required before they are deleted.
3. In the Maximum Size text field, enter the maximum number of log entries that can be stored on a server.
4. In the Maximum Age text field, enter a value that represents the amount of time a log entry is kept before being deleted. (Please refer to Administration Tool Date and Time Entry Formats in Chapter 2, Introduction for a list of valid values.)
5. Click Set Configuration to save changes, then restart the server.

Directing Email Error Reports

Server and Client Logs record, among other items, any errors that have occurred during the deployment process on either the client or server side. Since the timing of dealing with deployment errors is more important than viewing other log entries, DeployDirector can be configured to send an email report to any number of recipients whenever an error occurs.

The contents of an email error report is identical to that which is placed in a Client Log or Server Log. Even when error email reports are sent out, that information is still logged for future reference.

Recipients can be set by entering email addresses under the Error property node in the Administration Tool.

There are four types of errors, of which any combination can be sent to users:

client.local errors pertain to problems experienced by client machines.

client.connection errors pertain to problems experienced by client machines while connecting, or attempting to connect to a server.

server.local errors pertain directly to the server (e.g. the server was not able to read or write a file in the vault).

server.connection errors pertain to server-server communication problems.

Configuring Email Error Logging at the Cluster and Server Level

Email error logging can be set for the entire cluster, or particular servers, using the Remote Administrator. Whereas logging properties at the server level override the settings of the same property at the cluster level, server and cluster-level email error report settings are combined.

How cluster and server level email error logging properties are set and combined depends entirely on who in your organization is responsible for the maintenance of its deployment network.

As a general rule, if the scope of your deployment network is small enough that system administrators are responsible for all servers, email error logging properties should be set at the cluster level. However, if the geographic scope of your deployment network is regional, national, or even international, it is likely that different system administrators are responsible for maintaining different servers. If this is the case, error logging properties should be set at the server level; each individual server is configured to send email error reports to the appropriate system administrator.

To set up email error logging, you will need to enter outgoing mail server information, as well as recipient email addresses.

Setting email error recipients at the cluster or server level

1. In the Remote Administrator, navigate to the Cluster: Server Configuration: Error Emailing page, or the Server: Server Configuration: Error Emailing page, depending on whether the settings are meant to affect the entire cluster or the server to which the Remote Administrator is connected.

   For cluster-level settings, no matter which server the Remote Administrator is connected to, any saved changes made at the cluster level are replicated to all other servers that are part of the cluster.

2. In the "From" Account: Username text field, enter the login user name for the mail server.

3. In the "From" Account: Password text field, enter the login password for the same mail server.

4. In the "From" Account: Server text field, enter the name of the outgoing mail server.

   Once the outgoing email server properties are configured, a recipient node must be created for every person who is supposed to receive a copy of generated email error reports.

5. In the "To" Account: Address text field, enter all of the recipients full email addresses, separated from each other with spaces.

6. Select the appropriate Notification Levels check boxes, which represent the types and levels of error emails you want the recipient(s) to receive.

7. If necessary, repeat steps 5 and 6 for other groups of users who require different notification settings.

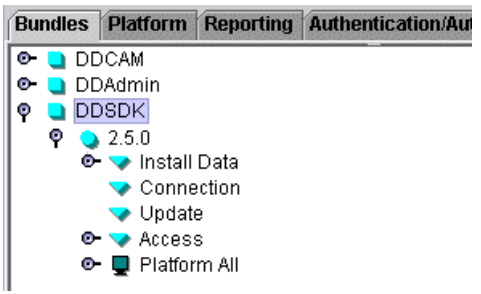
8. Click Set Configuration to save changes, then restart the server.

Any properties set at the cluster level will be replicated across the cluster.

Chapter 11

Customizing Functionality with the SDK

Many sites find that the DeployDirector Administration Tool provides all the control they need for the distribution and update of their applications. However, some capabilities are only possible by making changes to an application (such as adding a “Check For Updates” item to your Help menu) or by replacing some of DeployDirector’s own functionality (to use your site’s authentication system, for example). This is where the DeployDirector SDK comes in.



The DeployDirector SDK is provided for software developers and build personnel. It contains classes, source code, examples, and applications that enable developers to access DeployDirector from their applications and extend DeployDirector’s authentication, authorization, and security modules.

This chapter assumes the reader is familiar with DeployDirector concepts and terminology.

Deploying the SDK Files to Your Workstation

The DeployDirector SDK is packaged as a bundle in the vault that ships with the product. To begin using it, deploy it to your local workstation.

With the server running, the SDK can be deployed from either the DeployDirector Administrator's Page, or by entering a URL in a browser, for example:

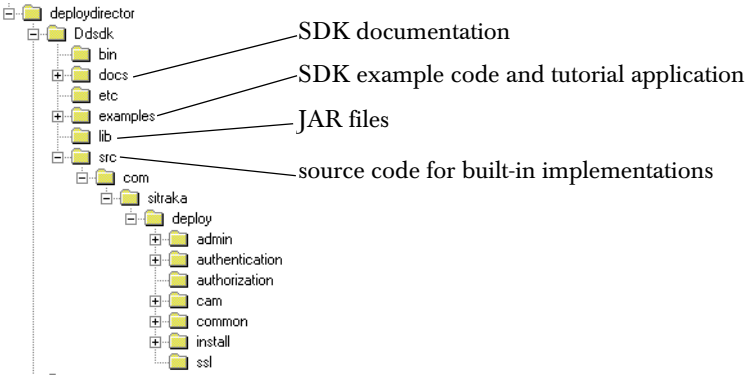
http://[your host name]:8080/servlet/deploy/ddsdk/install

(Replace "[your host name]" with the name of the machine running the DeployDirector server; the URL may also vary depending on your configuration.)

When prompted, enter your admin user name and password ("ddadmin" and "f3nd3r" by default). Follow the install applet prompts to specify where to install the SDK files on your system.

Overview of SDK Components

When the install has finished, the files and directories shown below are available on your computer.



Conceptually, the SDK consists of the components described in the following sections.

Client Application Classes (ddcam.jar)

A Java class library containing GUI components and other classes is provided for developers to use in their applications. DeployDirector features available to developers include:

- providing an "update application" GUI for users
- automatically checking for and selectively updating an application at any time during its execution, complete with user authentication and version authorization
- responding to Install and Server events triggered by DeployDirector
- automatically updating non-Java parts of the application

ddcam.jar is located in DDsdk\lib. End-user classes are located in the `com.sitraka.deploy` package.

Sample code is located in DDsdk\examples.

Documentation is located in this chapter and DDsdk\docs\api; can be accessed on Windows using the Start menu group created for the SDK.

Getting Started with the Client Application Classes

Like any third-party code library, a few basic steps are needed before you can add DeployDirector capabilities to your application:

1. Add the ddcam.jar file to your application's CLASSPATH.
2. Import the classes to be used by your application, for example:
   import com.sitraka.deploy.*;
   import com.sitraka.deploy.authentication.*;
3. Write code that implements the features described in this chapter in your application, using the documentation in this chapter and the Javadoc API reference documentation for guidance.

Testing and Debugging Note: Because most of the features provided by the client application library involve communication between the application and a running DeployDirector server and CAM, testing and debugging can be a bit tricky to set up. We recommend installing the standalone server on your local workstation, using the administration tool to create a test bundle for your application, deploying it to your workstation, and running it that way. This enables you to verify that your application's interaction with DeployDirector is working correctly.

Copy of SAM JAR (ddsam.jar)

A copy of the server-side application manager (SAM) classes is provided for developers to use when a site-specific authentication, authorization, or security module is needed. Using the source code and documentation provided, you can develop a unique module and add it to the DeployDirector SAM.

ddsam.jar is located in DDsdk\lib.

Source code for built-in modules is located in DDsdk\src.

Documentation is located in this chapter and DDsdk\docs\api; can be accessed on Windows using the Start menu group created for the SDK.

Getting Started with the SAM JAR

Adding site-specific modules to the SAM is a fairly involved process. Use the source code as a basis for your subclassed module. It, along with the documentation in this chapter and the Javadoc API reference, provides guidance on this process.

SDK Java Packages and API

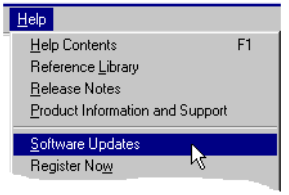
The following table summarizes the Java packages that are part of the SDK:

Package Summary	
`com.sitraka.deploy`	Provides classes and interfaces for adding deployment, updating, and version-checking functionality to any application. Also contains interfaces needed for custom authentication, authorization, and security modules.
`com.sitraka.deploy.authentication`	Contains classes that perform any user authentication needed by an application (both client and server side). Source code is provided as examples and to use as a base for subclassing to create new authentication modules.
`com.sitraka.deploy.authorization`	Contains classes that determine which products and versions a user is authorized to use. Source code is provided as examples and to use as a base for subclassing to create new authorization modules.
`com.sitraka.deploy.ssl`	Code that performs secure transmission of user authentication information (and even of deployed bundles). Included primarily as examples for users to replace with site-specific secure-socket implementations. Applications should not use these classes directly.

Adding Update Checking To Applications

For many applications and types of users, system administrators want to control precisely when and how users update their applications. Bundle configuration properties accessed in the administration tool allow them to do this.

However, end-users of your applications often appreciate the ability to check for new versions and upgrade the application at their own convenience. Popular or commercial software often provides automatic features as part of the application. DeployDirector enables you to provide this kind of feature to users of your applications very easily.



A Typical User-Update Feature provided in the Help Menu

The most common way to provide this feature to users is by adding a “Check for Updates” item to the help menu of the application.

Tip — You can avoid changing your code but still provide end-users with the ability to initiate update checks. When creating the bundle, you can add a shortcut that checks for updates (**Install Data | Shortcuts** node in the administration tool). This isn’t as visible as a menu item right within the application however.

DeployDirector provides three classes that provide this feature, each useful for a particular situation. All are located in the `com.sitraka.deploy` package.

CAMMenuItem and CAMJMenuItem are menu item subclasses that have built-in actions defined that access the DeployDirector CAM to check for updates to the application.

CAMAction is a Swing Action object that can handle update checks invoked from either menu items or toolbar buttons.

Important Note for Bundles

Because adding this feature changes how updates can occur, you should consider adjusting some of the properties in the bundle. By default, DeployDirector initiates checks and updates automatically.

You can use the administration tool to change the connection and update behavior of the bundle. For example, to leave the onus on users to check for and update their application, set the **Connection | Connect to Server** property to “User Initiated”, and the **Update | Policy** property to “Optional”.

CAMMenuItem and CAMJMenuItem Classes

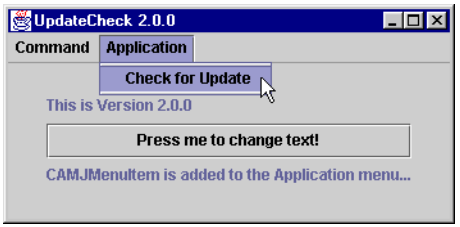
`CAMMenuItem` is an AWT-based MenuItem, and `CAMJMenuItem` is a Swing-based JMenuItem. This makes it easy to add them to any menu. Once added to a menu in your application, the component's event-handling mechanism uses the CAM to query the DeployDirector server when the menu item is selected by a user.

The following code fragment shows how to add this item to an AWT application menu:

```
import com.sitraka.deploy.CAMMenuItem;

Menu application_menu;
CAMMenuItem updateCheck_item;
...
// Add CAMMenuItem to the Application Menu
updateCheck_item = new CAMMenuItem();
application_menu.add(updateCheck_item);
```

Example code that uses the `CAMMenuItem` and `CAMJMenuItem` classes is included in the SDK, located in the DDsdk\examples\updatecheck directory (specifically versions 1.0.0 and 2.0.0).



To run or test these examples, you need to create bundle versions using these source files. Then deploy the program and run it.

CAMAction Class

The `CAMAction` class provides a general way to check for updates from any menu item or toolbar button. `CAMAction` implements the Swing Action interface. This mechanism is particularly useful when you want to provide more than one way to check for updates within your application.

The following code fragment demonstrates using one `CAMAction` instance to handle update-checking from both a menu item and a toolbar button:

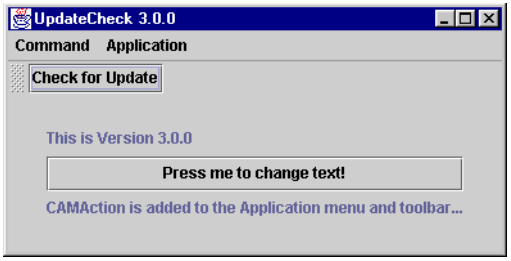
```
import com.sitraka.deploy.CAMAction;

JMenu application_menu;
JMenuItem button_item;
JToolBar tool_bar;
CAMAction cam_action_item;
...
 //create the Application Menu
 application_menu = new JMenu("Application");
 menu_bar.add(application_menu);

 //add a MenuItem for CAMAction
 cam_action_item = new CAMAction();
 application_menu.add(cam_action_item);

//create the toolbar
tool_bar = new JToolBar();

 //add CAMAction to the toolbar
 tool_bar.add(cam_action_item);
```



Example code that uses the `CAMAction` class is included in the SDK, located in the `DDSDK/examples/updatecheck` directory (specifically versions 3.0.0 and 4.0.0).

Advanced Update Checking for Applications

The `CAMAccess` class enables an application to implement version checking and updates in a completely different fashion than that provided by DeployDirector's menu item or `CAMAction` classes. Using `CAMAccess`, an application can:

- check for updates on its own, without being initiated by the end-user
- present its own "updates" dialogs and user interface for end-users
- implement specific update-handling for an application (such as never to allow a user to update to a ".0" release (such as "2.0.0"))

The methods in `CAMAccess` cover every step in the update process. The ones you need to use depend on what you're trying to do. The following table groups the methods in `CAMAccess` in the general order you might use them when implementing a new update mechanism in your application.

Stage	CAMAccess methods
Initialization	getErrorStream() isApplicationRunning() isGUI()
Checking for Update	getServerList() listRunningCAMS() checkForUpdates()
Controlling the Update	getCurrentVersion() getAuthenticationObject() queryUserAboutInstallingUpdate() queryUserToSelectVersion() getUpdatePolicy() getUpdateType()
Updating the Application	updateBundle() updateBundle(version)

Other Useful CAMAccess Methods

The `CAMAccess` class also provides methods that are not related to checking or updating applications. For example, the `displayDocument` method enables your application to display a web page in the system's default web browser. This can be useful for release notes or even for quick online documentation for your application.

Please see the Javadoc API reference for complete details on using the `CAMAccess` class.

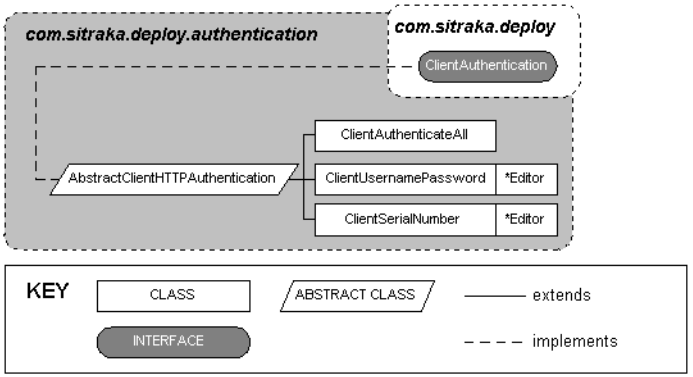
Client-Side and Server-Side Authentication

Authentication is the process of determining whether a user is who they claim to be. Authentication is the first part of the process of controlling access to DeployDirector bundles (authorization, described in the next section, is the second part).

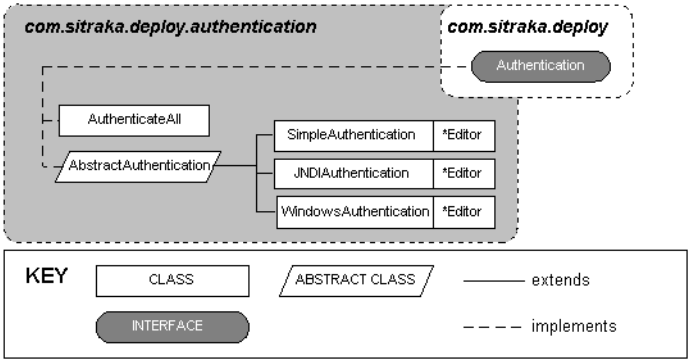
Programmatically, authentication and authorization have been modularly implemented in DeployDirector, allowing you to plug in classes whose properties best match your deployment environment and needs.

For more background information on authentication and authorization, including a description of the modules/classes, please see An Overview of User Authentication and Authorization in Chapter 9.

The following diagram provides an overview of the classes that perform client-side authentication in DeployDirector.



The following diagram provides an overview of the classes that perform server-side authentication in DeployDirector.



Custom Authorization Modules

Once a user has been authenticated, DeployDirector *authorizes* that user to access *particular bundles and versions*. Two basic authorization systems are provided with DeployDirector – one that authorizes anybody to access any version of any bundle, and one that authorizes users to only access the bundles specified in an internal data file (the administration tool is used both to specify the authorization method, and to enter user/bundle/version authorization data).

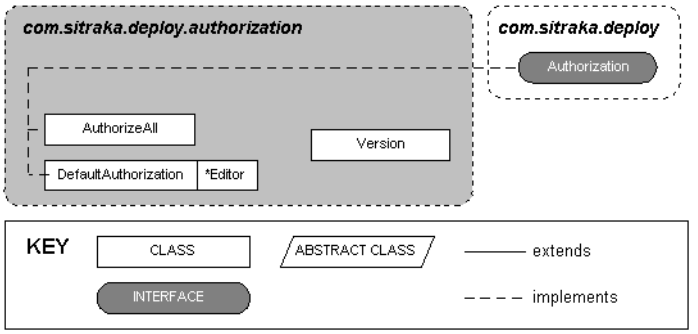
This system works well for most sites. Unlike authentication (which accesses external standard user account systems), authorization data is typically specific to DeployDirector, and does not generally access external site-specific databases. However, as DeployDirector becomes more widely-used at a site, it may be advantageous to store and access authorization data centrally.

DeployDirector enables you to create and plug in a custom authorization module. Authorization set up is done for one vault, so large sites that maintain multiple servers and vaults would need to copy or duplicate authorization data. In this case it may be better to create a central database and use a custom authorization module to access it using JDBC.

For more background information on authentication and details on specifying authentication for bundles, please see the *DeployDirector Administrator's Guide* chapter on "Authentication and Authorization".

Overview of the com.sitraka.deploy.authorization Package

The `com.sitraka.deploy.authorization` package implements the authorization modules built in to DeployDirector.



All authorization modules implement the `com.sitraka.deploy.Authorization` interface, which defines the methods the SAM calls to administer (define and edit) authorization data, and to perform authorization requests made by a CAM.

The `AuthorizeAll` class is extremely simple – because it simply returns "AUTHORIZED" for any request, it neither uses any data file, nor provides authorization editing.

The `DefaultAuthorization` class is probably better to start with – it provides a simple but complete authorization system. It stores and manages authorization data in a flat text file, and provides a GUI editor to allow administrators to specify and edit authorization information in the administration tool.

The `Version` class encapsulates a number of useful conventions for specifying and managing versions.

The source code for all of the classes in the authorization package is provided in the DDsdk\src\authorization directory of where you installed the SDK. Javadoc API reference documentation is installed with the SDK, and is also provided later in this chapter as a convenience.

Creating a New Authorization Module

The first step in creating a custom authorization module is to choose whether to base it on `AuthorizeAll` or `DefaultAuthorization`. One of these approaches probably matches your needs more closely than the other.

Next, consider whether to extend and override one of the built-in modules, or start entirely from scratch (with your class implementing the `Authorization` interface). Subclassing a built-in module makes sense when you only need to change a small part of how one of the built-in modules work, for example, to use a database for authorization information. Creating a new module from scratch makes sense when you need something entirely different, such as entirely different set of rules for version specification.

In either case, use the source code for the built-in modules as a starting point.

Basic Authorization Assumptions

- Authorization modules assume any authorization requests are made by valid users, since authorization in DeployDirector takes place *after* user authentication.
- While there is no direct dependency on the user authentication used, the user ID is a piece of information common to both; the authentication and authorization methods should handle user IDs in the same way. For example, if your authentication method does not use user IDs (such as ClientAuthenticateAll), it does not make sense to use an authorization module that checks user IDs.
- Authorization functions take place in the SAM. There are two parts to an authorization module: handling authorization requests from a CAM (is this

user authorized to access this), and managing authorization data (adding user / version data in the administration tool).

Authorization-Handling Methods

The following table provides notes for implementing the `com.sitraka.deploy.Authorization` interface methods that handle authorization requests:

Method	Implementation Notes
isAuthorized(user_id, app_name, version)	Performs authorization — this is the key method, Examines the parameters specified, and determines whether it is authorized. Returns `AUTHORIZED` or `NOT_AUTHORIZED` (defined in Authorization interface). The user ID cannot be null. If null is specified for the application or the version, it is taken to mean "any".
usesDataFile setDataFile(data_file)	File handling methods. In the DefaultAuthorization module the user/application version data is stored in the file specified by data_file. The authorization module does not know the details of filename and location (this is handled by the SAM), so if your custom module uses a database, data_file should contain information your module will use to access the database. The `setDataFile` method is called for every authorization request, and it triggers reloading of the data, so take care to load the file efficiently.

User / Version Data Editing Methods

The following table provides notes for implementing the `com.sitraka.deploy.Authorization` interface methods that handle editing user / application version information in the administration tool:

Method	Implementation Notes
hasEditor getEditorComponent	Unless authorization data will be edited outside of the administration tool, you should provide a GUI editor. This editor allows administrators to add and change user / application version information. The authorization module should store and use one instance of the editor, rather than creating a new editor each time the `getEditorComponent` method is called.
isModified commitChanges	Tracks whether the data has actually changed. Use this to avoid sending the data file to the server every time the editor is invoked. The `commitChanges` method is called by the administration tool as a user edits user/application version data.

When authorization data is edited while the server is running, the GUI editor needs a way to get changes back to the authorization module. One way is to cause user actions in the editor to trigger a resynchronization. Another way is to have the editor and Authorization module share a data model (this is the method used by DefaultAuthorization).

Using Secure Socket Encryption

DeployDirector can automatically encrypt all data sent between servers and client desktops. This is useful in any situation where data may be transferred over a public network, such as the Internet. DeployDirector enables you to use virtually any third-party Java-based encryption technology, but no encryption technology is built into the product. This is because the transfer of encryption technology between countries is often controlled by government export regulations.

For more background on encryption in general, and DeployDirector's security features, please see:

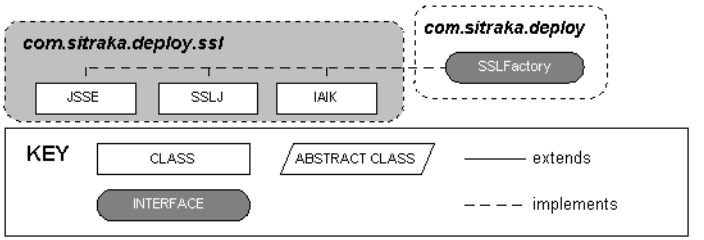
- RSA's Frequently Asked Questions about Today's Cryptography – http://www.rsasecurity.com/rsalabs/faq/
- The chapter on "Security and Integrity."

Support is fully implemented for Sun's Java Secure Socket Extension (JSSE) encryption. Starting implementations are provided as source code for other popular Java-based encryption libraries. This enables non-U.S. companies to use unrestricted encryption technology, or enables anyone to plug-in the SSL (secure sockets layer) library of their choice.

Note: It may not be necessary to use DeployDirector's encryption features. If your site's internal and/or external network gateways are set up to encrypt HTTP data where necessary, authentication and deployed application data will be encrypted the same as any other HTTP data over these gateways.

Overview of the com.sitraka.deploy.ssl Package

The `com.sitraka.deploy.ssl` package contains support for the encryption systems that DeployDirector can use.



These can be thought of as "bridge" classes that enable you to "plug-in" a particular Java-based encryption system. All of these classes implement the `com.sitraka.deploy.SSLFactory` interface, which defines the methods DeployDirector calls to work with the third-party encryption system.

Using these classes, you can implement one of the supported encryption systems for your application. It is also possible to write your own “bridge” class to use a site-specific encryption system.

Using JSSE, SSL-J or IAIK Encryption

To use one of the encryption systems supported by DeployDirector, you need to add the particular SSL library to both the CAM and SAM, and specify the name of the “bridge” class using the administration tool. The following describes this process in detail.

SAM-side:

Either combine the ddsam.jar with the third-party SSL library JAR, or add the SSL JAR to the Classpath of the server. Restart the server and use the administration tool to do the following:

1. Specify the name of the “bridge” class by editing the deploy.http.sslsocketfactory property located on the http node of the Server tab. Example settings:
   com.sitraka.deploy.ssl.JSSE
   com.sitraka.deploy.ssl.SSLJ
   com.sitraka.deploy.ssl.IAIK

CAM-side (using the administration tool):

1. Create a new version of the "ddcam" bundle.
2. Add the third-party SSL library JAR files to the CAM bundle's Classpath object (located on the Platform All > Java > Classpath node of the Bundles tab).
3. Specify the name of the "bridge" class by adding the following System Property (located under the Install Data node on the Bundles tab):
   Name: deploy.http.sslsocketfactory
   Value: name of bridge class, for example com.sitraka.deploy.ssl.JSSE

In all cases, the prerequisite is to obtain the SSL library from the third-party vendor.

Using a Site-Specific Encryption System

In some cases you may want DeployDirector to use an entirely different SSL library. To accomplish this requires creating a class that provides a "bridge" between the third-party SSL library, and DeployDirector. Creating a "bridge" class that DeployDirector can use to access your site's encryption system is fairly straightforward. The source code for the default SSL systems is provided in the DDsdk\src\ssl directory; use this as a starting point.

The key to creating an implementation that will work with DeployDirector is to implement the `com.sitraka.deploy.SSLFactory` interface, which defines the methods DeployDirector calls to work with the third-party encryption system.

The class you create must provide a no-argument constructor.

Once created, add the third-party SSL library and your "bridge" class as described in "Using JSSE, SSL-J or IAIK Encryption" on page 225.

Notes on Firewalls, Proxies, and SOCKS DeployDirector can use encryption with firewalls, as long as the SSL implementation used has a constructor that takes an already-established socket, or the bridge class implements the `startSSLHandshake()` method. This is because DeployDirector first creates its own connection, possibly going through SOCKS or HTTP proxies, and only then invokes the SSL handshake. If the SSL implementation does not provide one of these methods, the connection will not work through proxies.

Certificates and SSL in Java SSL connections require certificates to validate the remote site. You may need to install other root certificates into the JRE you are using in order to use root certificates from other signers (such as Entrust). The documentation for "keytool" describes how to do this, located at http://java.sun.com/products/jdk/1.2/docs/tooldocs/win32/keytool.html .

A

B

C

D

E

F

H

I

J

L

V

W