INTERWOVEN®

OpenDeploy® Administration Guide

Release 5.6

INTERWOVEN

Interwoven, Inc.
803 11th Ave.
Sunnyvale, CA 94089
http://www.interwoven.com
Printed in the United States of America
Release 5.6
Part # 90-00-20-21-00-560-300

Table of Contents

About This Book

OpenDeploy Administration Guide is a guide to install, configure, and use OpenDeploy ®. It is primarily intended for webmasters, system administrators, and those involved in deploying content between development servers and production servers.

If you are using OpenDeploy in conjunction with TeamSite®, you should also know TeamSite functionality and terminology. Many of the operations described in this manual require *root* or *Administrator* access to the OpenDeploy host server. If you do not have root or Administrator access to the OpenDeploy host server, consult your system administrator.

This manual uses the term "Windows" to indicate any supported version of the Microsoft Windows operating system, such as Windows NT® or Windows® 2000.

This manual uses the term "UNIX" to indicate any supported flavor of the UNIX® operating system.

Windows: Users should be familiar with either IIS or Netscape® Web servers, and with basic Windows server operations such as adding users and modifying Access Control Lists (ACLs).

UNIX: Users of this manual should be familiar with basic UNIX commands and be able to use an editor such as emacs or vi.

It is also helpful to be familiar with regular expression syntax. If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

Notation Conventions

This manual uses the following notation conventions:

Convention	Definition and Usage
Bold	Text that appears in a GUI element (for example, a menu item, button, or element of a dialog box) and command names are shown in bold. For example: Click **Edit File** in the Button Bar.
Italic	Book titles appear in italics. Terms are italicized the first time they are introduced. Important information may be italicized for emphasis.
`Monospace`	Commands, command-line output, and file names are in monospace type. For example: The `iwodstart` command-line tool starts an OpenDeploy deployment task.
`Monospaced italic`	Monospaced italics are used for command-line variables.For example: `iwodstart` *`deployment`* This means that you must replace *`deployment`* with your values.
`Monospaced bold`	Monospaced bold represents information you enter in response to system prompts.The character that appears before a line of user input represents the command prompt, and should not be typed. For example: **`iwodstart`**
`Monospaced bold italic`	Monospaced bold italic text is used to indicate a variable in user input. For example: **`iwodstart`** ***`deployment`*** means that you must insert the values of *`deployment`* when you enter this command.
`[]`	Square brackets surrounding a command-line argument mean that the argument is optional.

Convention	Definition and Usage
|	Vertical bars separating command-line arguments mean that only one of the arguments can be used.

Other OpenDeploy Documentation

In addition to this Administration Guide, OpenDeploy includes the following documentation components:

- *OpenDeploy Installation and Reference Guide*
- *OpenDeploy Release Notes*
- Online help

OpenDeploy Installation and Reference Guide

OpenDeploy Installation and Reference Guide is a manual that contains installation and host configuration information for OpenDeploy. It also contains reference material on topics such as configuration DTDs, command-line tools (CLTs) and ports. Use this manual to find information on a specific reference topic quickly and easily.

OpenDeploy Release Notes

OpenDeploy Release Notes contains supplemental and late-breaking information regarding OpenDeploy not found in the other documentation. Refer to the *OpenDeploy Release Notes* for information regarding supported platforms, installation requirements, new features and enhancements, and known issues.

Online Help

OpenDeploy includes online help topics associated with each window in the OpenDeploy user interface. You can access the associated help topic by clicking the Help link in the upper right-hand corner of the window. The help topic will appear in a separate browser window that you can move and resize. You can display a navigation panel that provides you access to all the help topics by clicking the Show Navpane button. This panel provides you the ability to access help topics through the contents, index, and by using keyword searching.

Chapter 1

Introduction to OpenDeploy

Welcome, and congratulations on selecting the Interwoven OpenDeploy software! OpenDeploy is an industry-leading content distribution product for deploying static and dynamic enterprise content, including Web sites, code, and documents, to a multi-tier, multiple server environment. OpenDeploy runs on a variety of common servers, and is well-suited for a cross-platform enterprise.

OpenDeploy is a key part of your enterprise content management infrastructure, along with software products such as TeamSite, TeamSite Templating, DataDeploy™, and OpenSyndicate™. Using OpenDeploy in conjunction with these products provides you added features and flexibility otherwise not available. However, you can also elect to use OpenDeploy separately from such products for moving your enterprise content files.

OpenDeploy is based on advanced technology providing enterprise content transfer capability based on the following principles:

- Performance — OpenDeploy maintains a high level of transaction speed for large deployments, such as those with tens of thousands of files. Compression of deployed content is also supported as an option.
- Scalability —You can add additional hosts easily. Your ability to deploy content to multiple production servers simultaneously is greatly enhanced.
- Ease of use — A browser-based user interface allows easy configuration and use. Installation is also simplified, as no client software installation is required for you to operate OpenDeploy remotely from the actual OpenDeploy server. All you need to access OpenDeploy is a supported browser.
- Transactional capability — OpenDeploy supports transactional deployments, which will restore a server host receiving deployed files back to its previous existing state in the event the deployment is unsuccessful.

- Security — Defined user and administrator deployment roles ensure that individuals can only perform the tasks for which they are authorized. Encrypted deployments protect against unauthorized access to files.
- Flexibility — OpenDeploy supports the use of external scripts through its Deploy and Run feature.

Task Checklist for Using OpenDeploy

This section provides a checklist of key tasks you must perform to install, configure, and use OpenDeploy.

- Read this chapter in its entirety before proceeding to the next step. Here you will find an overview of the OpenDeploy product, how it works, and how you can best use it to meet your organizational needs.
- Install the OpenDeploy software components on a single or multiple server hosts. Refer to "Installation" on page 22 in the *OpenDeploy Installation and Reference Guide* for more information.
- Configure your bootstrap administrator. This is required for you to log on to OpenDeploy for the first time. Refer to "Configuring the Bootstrap Administrator" on page 51 in the *OpenDeploy Installation and Reference Guide*.
- Define the targets in your host's nodes configuration file. Each target host must have the base server or receiver software installed on it. Refer to "Defining Target Host Nodes" on page 57 in the *OpenDeploy Installation and Reference Guide*.
- Modify your OpenDeploy host's base server configuration file to allow incoming deployments from other OpenDeploy hosts. You can also configure settings regarding your host's logging, scheduling, encryption, and other features. Refer to "Modifying the Base Server Configuration File" on page 62 in the *OpenDeploy Installation and Reference Guide*.
- Modify the receiver configuration files of any target hosts that have the receiver software installed on them. Refer to "Modifying the Receiver Configuration File" on page 83 in the *OpenDeploy Installation and Reference Guide*.
- Start the OpenDeploy services or daemons, which provide the functionality of the product. See "Starting OpenDeploy" on page 47.

- Access the OpenDeploy user interface. See “OpenDeploy User Interface” on page 62.
- Add the OpenDeploy hosts that you want to view and use to the user interface. See “OpenDeploy Host Management” on page 65.
- Add Administrator and User roles to those who will administer OpenDeploy host servers and run specific deployments on them. See “Roles and Authorization” on page 75.
- Familiarize yourself with how to start a deployment. See “Starting a Deployment” on page 87.
- Perform and monitor a test deployment on your OpenDeploy host using the default configuration settings. See “Performing a Test Deployment” on page 90.
- Familiarize yourself with how to perform a simulated deployment. This feature is useful for seeing how a deployment would take place without actually moving the files. You can also use it to determine if files on the target host are different from those on its source host, which can be indicative of accidental or intentional modification of content. See “Performing a Simulated Deployment” on page 91.
- Familiarize yourself with OpenDeploy deployment scheduling. See Chapter 5, “Scheduled Deployments” on page 223.
- Familiarize yourself with OpenDeploy logging. See Chapter 6, “Logging” on page 243.
- Familiarize yourself with OpenDeploy reporting features. See Chapter 7, “Reporting” on page 265.
- Familiarize yourself with how OpenDeploy deployment configurations work and are structured. Although you can create and edit deployment configuration files using the Deployment Configuration Composer, you should also know how the configuration files themselves are structured, and how to make changes in the files using a text or XML editor. See “Deployment Configuration Files” on page 107.
- Modify a sample deployment configuration to work on your OpenDeploy host and perform a practical deployment of files. See “Using Example and Sample Configurations” on page 161.

- Familiarize yourself with the OpenDeploy deployment types:
  - Directory comparison, where files on a source and target file system directory are compared and the differences are deployed. See “Directory Comparison Deployments” on page 124.
  - File list, where a list of files on the source list determines which files are deployed. See “File List Deployments” on page 135.
  - TeamSite comparison (if you are using OpenDeploy in conjunction with TeamSite), where files on two TeamSite areas are compared and the differences are deployed. See “TeamSite Comparison Deployments” on page 128.
- Familiarize yourself with how to define target host-based overrides of deployment configuration settings. This allows you to modify how files are deployed on a target-by-target basis. This is useful when deploying files to multiple targets. See “Defining Target-Based Overrides” on page 141.
- Familiarize yourself with specialized deployments, such as:
  - Transactional deployments, which will roll back a deployment and restore the target hosts to their previous states if a deployment is unsuccessful for any reason. See “Transactional Deployments” on page 144.
  - Fan-out deployments, which deploy a set of files to several target hosts simultaneously. See “Fan-Out Deployments” on page 145.
  - Multi-tiered deployments, which allow a deployment to be deployed across a series of OpenDeploy sender hosts and their respective target hosts. See “Multi-Tiered Deployments” on page 150.
  - Reverse deployments, which allow files that are added or modified on a target host to be deployed back to the sending host. See “Reverse Deployments” on page 156.

  OpenDeploy includes sample files of these deployments that you can review and modify for your own use. See “Using Example and Sample Configurations” on page 161 for more information.
- Familiarize yourself with other OpenDeploy features as desired. See “Deployment Features” on page 171.
- Familiarize yourself with how to compose and edit deployments using the Deployment Configuration Composer. This is a tool in the OpenDeploy user interface that allows you to create new or edit existing deployment configurations more easily and with less risk of error than by modifying the deployment configuration file using a text or XML editor. See “Composing Deployments” on page 313.

Deployment Planning Considerations

The following section poses questions regarding common planning issues. It then describes how OpenDeploy can meet those issues.

How is content being distributed in your company?

Is there a standardized method for moving content from the development server or hub to the production servers? There are a variety of open and proprietary products for doing this task, but they have varying levels of reliability and support. Additionally, many are platform-specific and are ill-suited for a large enterprise with a combination of Windows and UNIX servers.

Are you distributing the content manually?

In many enterprises, the task of moving enterprise content files from their development servers to their destinations requires an employee to manually move the files. There is little or no automation, and the risk of user error exists each time files are moved. There is also no scheduling capability, requiring the deployment administrator to be present when the deployment occurs.

Do you have more than one destination server or multiple data centers?

Larger enterprises often have multiple servers and data centers. Some might be mirror sites requiring each server or data center to store the exact same files. In other cases, multiple data centers might have a portion of their content customized to their particular audiences, for example, different geographical regions.

How do you synchronize multiple servers with new or updated content?

An enterprise with multiple data centers or mirrored sites needs the ability to synchronize its servers to contain the same content. A high level of precision might also be required, such as if a production server were being updated at a specific time to coincide with a product release.

Do you have any legal compliance that requires you to prove what content was deployed on a particular day?

Legal compliance of content is required by many companies that conduct business on the Internet or in regulated industries. An enterprise must be able to create and archive a snapshot of their content at a particular point of time, and even be able to "roll back" servers to an earlier version to an earlier version of content.

Case Study: Acme Corp.

In this section we introduce a fictional company named *Acme*, headquartered in San Francisco with regional data centers in New York, London, and Tokyo. There are a mix of Windows and UNIX development and production servers.

Currently, each regional data center is responsible for receiving content from the home office in San Francisco, making any additions or changes, and moving the files to their regional production servers. Because of rapid growth and the acquisition of smaller companies, no single deployment solution exists. Instead, different groups within the enterprise use their own individual deployment solutions. Here is how each regional component moves files:

- San Francisco — locally scripted implementation of FTP
- New York — third-party deployment software with minimum features
- London — rsync
- Tokyo — deployment software for UNIX platforms only

Each of these solutions has limitations in the areas of support, cross-platform compatibility, and scalability. Additionally, each solution requires a separate knowledge base and level of expertise. If Acme could standardize on a single deployment solution that met the needs of each regional component, but also ensured reliable support and a common knowledge base, deployment of enterprise content would be much easier to execute and manage.

Currently, deployments are performed by a small number of administrators with a keen understanding of how the deployment software works. In some cases, there is no scheduling capability, requiring one of these administrators to be present whenever a deployment takes place. Deployments typically occur after regular work hours to avoid slowing the network with a high level of traffic. As a result, the deployment administrator must work non-standard hours to oversee the deployment. If no administrator is available, the deployment cannot take place.

Many of the deployments send the same content to multiple sites. For the existing solutions with no multiple target capability, the same deployment procedures must be performed serially for each recipient server. This is not only a time-consuming and tedious task, but it increases the risk of a user or system error that can compromise the deployment. If the recipient servers include one or more mirror sites of a production server, it is vitally important that all of these servers contain the exact same content. It is preferable that the deployment not go out at all if it is not guaranteed that all sites will receive the new files simultaneously.

The amount and size of the deployed files themselves can vary. For example, Acme's Web sites are updated frequently. In some cases, the update requires the addition of only a few files. In other cases, the entire Web site needs updating, requiring the deployment of thousands of files. Ideally, Acme's deployment solution should be able to determine what files need to be deployed for each situation, rather than simply redeploying an entire set of files every time.

Based on this scenario, Acme needs the following features and capabilities from its deployment solution:

- A single solution that can be used by all the data centers — a single solution greatly simplifies issues of upgrades, support, and training.
- Full software support and training — reduce the pressure on the Acme staff to teach themselves the product and attempt to support or troubleshoot it.
- Cross-platform capability — require the software to not only run on different platforms, but also to be able to send and receive files from servers of differing platforms.

- Multi-target deployment capability — being able to send a deployment to multiple targets simultaneously speeds and simplifies transferring data from development servers and data center hubs to their destinations. User and system errors are also reduced.
- Multi-tiered deployment capability — allow multiple OpenDeploy servers to redeploy files to multiple tiers of servers. This feature allows the "chaining" of deployments from one tier to the next. For example, a file set could be deployed from the development server to a data center, and then redeployed to multiple production servers automatically.
- Scheduling of deployments — allow deployments to take place during periods of lower network traffic, such as after work hours or on the weekend. Deployments can be scheduled to coincide with other activities, such as a new project launch. Employees do not have to be present during the deployments, freeing them from working odd hours and allowing them to concentrate on other activities.
- Determination of deployable files — be able to identify the delta between a like set of files on the source and target hosts. This process identifies only those files that require deployment, avoiding deploying the identical or unnecessary files.
- Mobility of deployment administration — be able to administer and monitor deployments from multiple locations throughout the enterprise remotely.
- An reporting structure that produces summary deployment reports.

The OpenDeploy Advantage

OpenDeploy can meet Acme's requirements in the following ways:

- OpenDeploy is an established product with full documentation, technical support, and training to back it up.
- OpenDeploy provides an automated solution based on customizable configurations. After you set up your deployment configurations, you can schedule them to occur whenever you want, without an administrator present.
- Each deployment follows the exact specifications of the configuration, eliminating the possibility of user error when moving files. You also can relieve employees from having to move files manually, and redirect their efforts to other tasks.
- OpenDeploy can perform a comparison of the files residing on the source server and the recipient targets, or between two TeamSite areas. Only those files that meet the deployment criteria are deployed, saving time and network resources.
- OpenDeploy allows you to deploy a set of files from a single source host to any number of recipient hosts. If you have multiple tiers of servers, for example development servers, hubs, and production servers, you can use multiple OpenDeploy servers to redeploy files from one tier to the next.
- OpenDeploy provides IT managers a high level of flexibility in assigning and managing administrative and user role access to deployments. Specific individuals can have access to starting and scheduling specific deployments based on departmental needs without allowing undo access or burdening IT staff.
- OpenDeploy uses a browser-based interface accessible from any computer within the enterprise for many deployment and monitoring tasks, allowing deployment administrators to start and monitor deployments from remote locations.
- OpenDeploy can deploy from TeamSite staging areas, editions, and workareas to file system locations on the target hosts. For example, you can redeploy legacy TeamSite editions in case you want to "roll back" a Web site to a previous version or recreate a past Web site.

The OpenDeploy Environment

OpenDeploy consists of a suite of interlocking services that create the OpenDeploy environment. Within the OpenDeploy environment are the following components:

- Base server software — enables the host to start deployments to other hosts, as well as to receive files deployed from other OpenDeploy hosts.
- Receiver software — enables the host only to receive deployed files.
- Administration server software — manages the following OpenDeploy functions:
  - Generation of the browser-based user interface.
  - Access to OpenDeploy hosts, features, and functions, based on user and administrator roles, through the operations server software.
  - Reporting through the reporting server.
- TeamSite (optional) — OpenDeploy includes optimizations for deploying files from selected TeamSite staging areas, editions, and workareas to file system locations on the target hosts.

Base Server

The core of the OpenDeploy environment is the *base server.* A base server is a host with the base server software installed and configured on it. A base server can both send and receive deployed files. When the base server deploys files to another host, it assumes the role of the *source host* in the source/target relationship. If the base server itself is receiving deployed files, it becomes the *target host*.

The base server can be a development server within the enterprise firewall, or it might be a hub outside the firewall responsible for redeploying files it received to another set of target hosts. The number and positioning of base servers in the OpenDeploy environment is determined by the deployment requirements.

The base server host is where the *deployment configurations* are located. Deployment configurations are XML-based files that determine how and to where a deployment will take place. The base server also maintains a variety of log files for each deployment and for general base server activities.

Base Server Versions

OpenDeploy base server software comes in the following versions:

- OpenDeploy — this is the full-featured version of the product.
- OpenDeploy Trial — this is the full-featured version with a 30-day time limit starting from the time of installation. OpenDeploy Trial is intended for those who want to try all the OpenDeploy features and functions on a limited-time basis before committing to purchasing the product.
- EasyDeploy — this is a version of OpenDeploy with the following limitations:
  - No fan-out deployments are allowed. Deployments limited to a single target file location on a single receiver host.
  - No multi-tiered deployments are allowed.
  - Encryption of deployments is unavailable.

Each OpenDeploy base server option is installed separately.

Receiver

A *receiver* is a host with the receiver software installed on it. A receiver host can only receive files as a target host in the source/target relationship. Receiver hosts are typically production servers outside the firewall that serve the deployed content to its intended audience but do not need to redeploy the content files any further.

Administration Server

The OpenDeploy administration server is responsible for managing and generating the browser-based OpenDeploy user interface, and also for managing the Administrator and User role access to OpenDeploy, such as the ability to create and start deployments. The administration server does not send or receive files itself, but works in conjunction with the source and target hosts. The administration server software can reside with all the other OpenDeploy components on a single server, or it can reside on a host separate from the base server or receiver software.

Operations Server

The OpenDeploy operations server provides the access management infrastructure for all the administrators and users within the OpenDeploy environment. If you are using OpenDeploy in conjunction with TeamSite 5.0 or later, you must use the TeamSite OpenAPI instead of the operations server software.

Reporting Server

The OpenDeploy reporting server publishes detailed summaries of events taking place on the base server and receiver hosts in the OpenDeploy environment. These summary reports can then be accessed through the browser-based user interface, by entering a command-line utility, or integrated into other reporting structures.

TeamSite

OpenDeploy can be tightly integrated with TeamSite, allowing you to manage and deploy your content within the same environment. In many cases, customers install their TeamSite and OpenDeploy base server software on the same server host. OpenDeploy can compare two TeamSite areas, such as staging areas, editions, and workareas, and deploy the differences to a file system location on the target host. OpenDeploy can compare two TeamSite areas and deploy only those files that are different.

How OpenDeploy Works

The OpenDeploy source host processes deployment configurations. These configurations determine the type of deployment being performed, as well as what other functions and features OpenDeploy will perform in the course of the deployment. The deployment configuration also specifies what target hosts will receive the deployed files.

Source-Target Relationship

A deployment of content files begins at the *source host* (the host sending the files) and ends at the *target host* (the host receiving the files).

Depending on the software you install on a server, an OpenDeploy host can act as both a source and target host, or as a target host only. Source hosts are not restricted in the number of target hosts available to them, providing the target host has the proper software installed and licensed. Similarly, a target host can receive deployments from any number of source hosts.

In most enterprises, a single source host will deploy files to many target hosts. Depending on the type of deployment taking place, the deployed files will either be exactly the same for all targets, or customized based on a comparison of file differences between the source host and each target host.

Source Host

The source host is a server with the OpenDeploy *base server* software installed. The source host originates and manages all deployments. The source host maintains a list of all target hosts to which it can deploy files. You can deploy files from one or more source file locations, which can be either file system locations or TeamSite areas.

Target Host

The target host is a server with the OpenDeploy base server or *receiver* software installed. Any such server in the OpenDeploy environment can receive deployed files, as long as the server is known to the source host and has declared itself available for receiving deployments from the appropriate source hosts.

Figure 1 shows the source/target relationship between a source host (with the base server software installed) and three target hosts (one with the base server software and two with the receiver software installed).

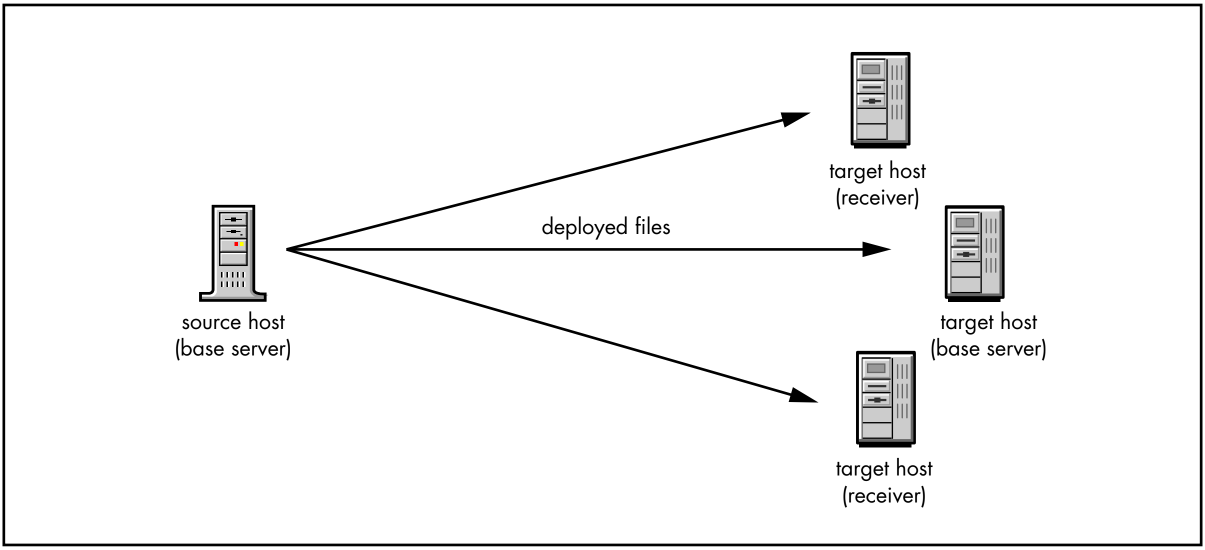


Figure 1: Single Base Server and Multiple Receivers

Those target hosts that have the base server software installed are capable of redeploying the files they receives to another set of targets. A deployment originating from a single source host might be redeployed several times to more and more targets automatically, saving time and labor, and ensuring content synchronicity among all the OpenDeploy hosts. In this case, the host receiving the original deployment is a target host, but when it redeploys those files to a new set of target hosts, it does so as a source host. This server must have the appropriate software installed and licensed to be able both to receive and redeploy files.

In Figure 2, one of the target hosts for the first deployment has the base server software installed, and can redeploy as a source host the files it received as a target host.

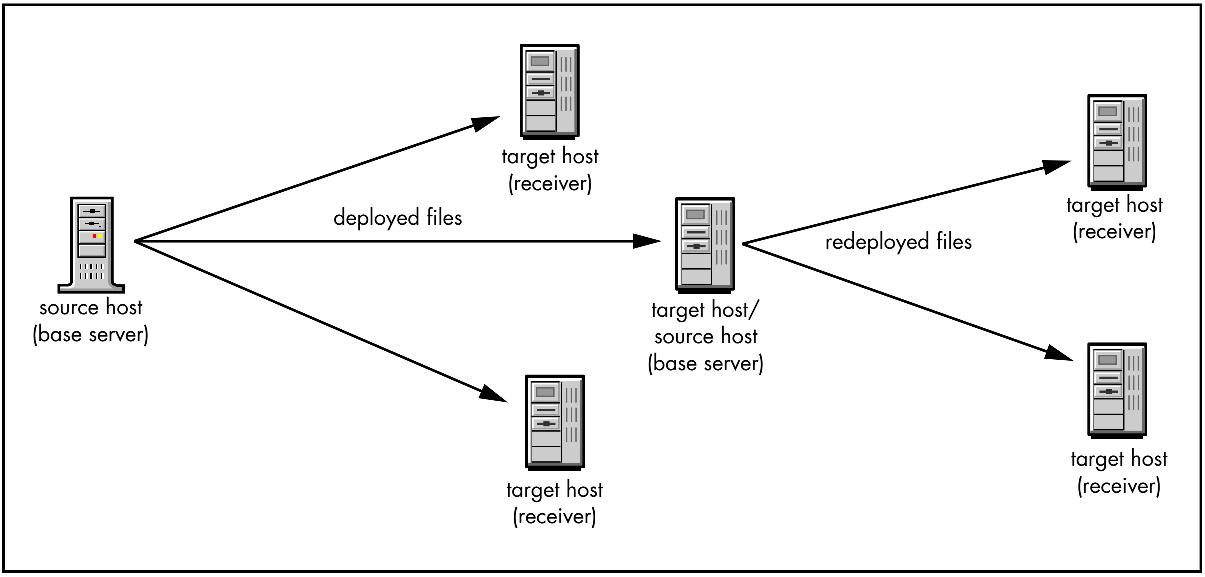


Figure 2: Redeploying Files Using Multiple Source Hosts

Deployment Configurations

The criteria for how to determine which files get deployed is specified within a *deployment configuration*. An OpenDeploy source host can process any number of distinctly named deployment configuration files, each of which can deploy files to different target area on different target hosts under different conditions. A typical deployment configuration will specify the following:

- Source file locations
- Target hosts
- Target file location
- Type of deployment
- Logging

- Use of external pre- and post-processing scripts
- Filters for limiting deployed files
- Rules for comparing files
- Rules for transmitting files
- Rules for setting file and directory permissions on deployed files

Figure 3 shows how a source host uses the information in a deployment configuration file to perform a deployment.

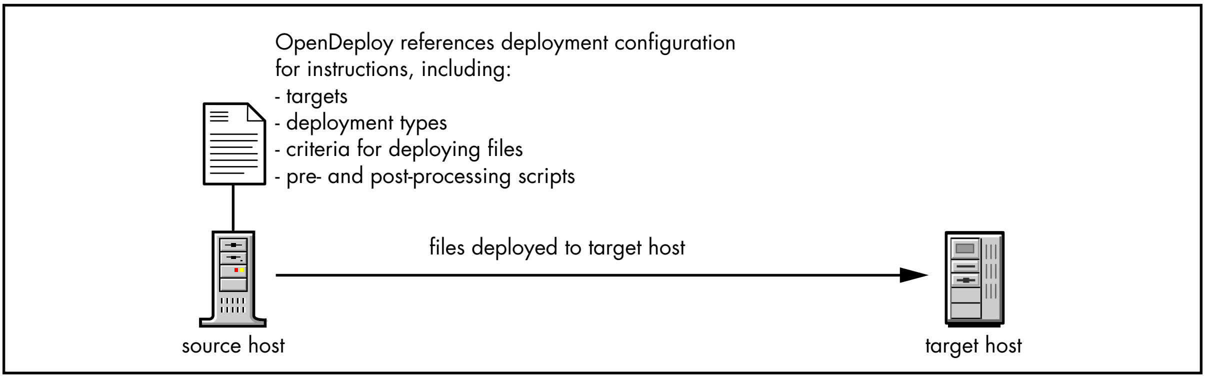

OpenDeploy references deployment configuration
for instructions, including:
- targets
- deployment types
- criteria for deploying files
- pre- and post-processing scripts
files deployed to target host
source host
target host

Figure 3: How OpenDeploy Uses Deployment Configurations

File Location Definitions

From the source host, one or more source file locations can be specified for deployment to a single target file location. The grouping of one or more source file locations deploying files to a target is known in the deployment configuration as a *definition*, and is the basis of any deployment. Each definition must have a unique name which you can assign to it, either by editing the deployment configuration file, or using the Deployment Configuration Composer.

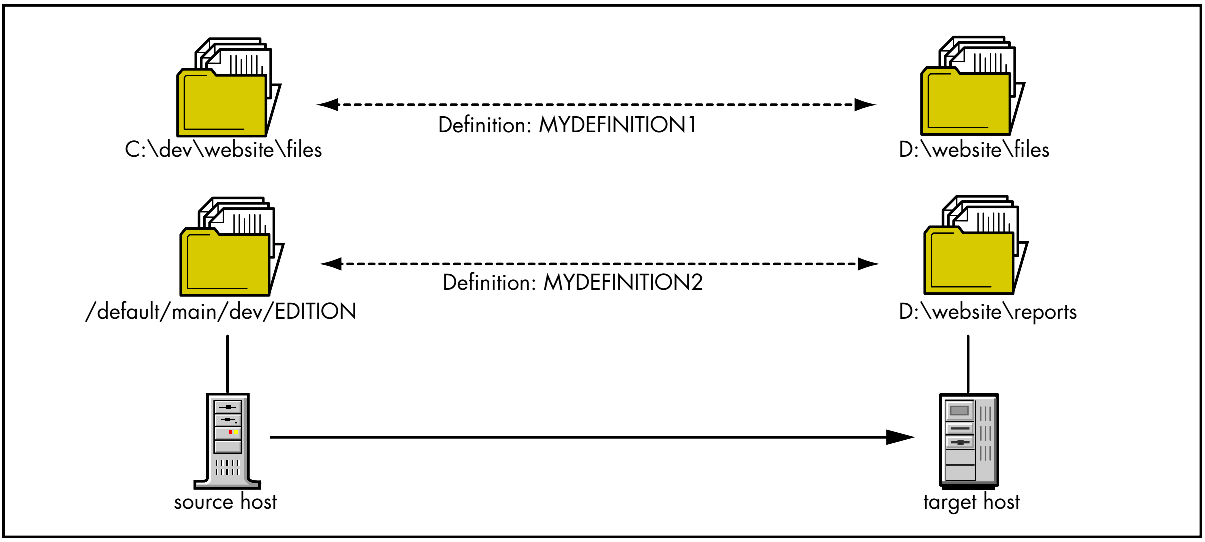


Figure 4: Definitions

Figure 4 shows how a single OpenDeploy host can deploy files from multiple source file locations (file system locations or TeamSite areas) to multiple target file locations on the same source and target host. Each definition can have one or more source file locations, but only a single target file location.

File Deployment Criteria

OpenDeploy determines file *deployment criteria*—whether a file or directory should be deployed from the source host to the target host—based on one of the following methods:

- Comparing files between file system locations on a source and target host and determining the differences
- Comparing two TeamSite areas on the source host with TeamSite installed and determining the differences
- Deploying files referenced in a list of files without comparing them

Comparison-Based Deployment Eligibility

A comparison-based deployment is the result of comparing one set of files with a second set of files. These file sets can be either file system locations on the source or target hosts, or a pair of TeamSite areas within the same backing store on a source host that has TeamSite installed. You can configure OpenDeploy to compare two TeamSite areas in any single backing store on a multi-backing store TeamSite host.

The default comparison criteria used to determine whether or not a given file should be deployed are:

- Modification date (when the source-side file is newer than its target-side equivalent)
- Type mismatch (a file and a directory sharing the same name)
- Size difference

In addition, OpenDeploy provides a variety of other deployment criteria you can use or ignore in your deployment configurations, including:

- Source-side file is older than target-side equivalent
- Access control list (ACL) difference (Windows only)
- User difference (UNIX only)
- Group difference (UNIX only)
- Permission difference (UNIX only)

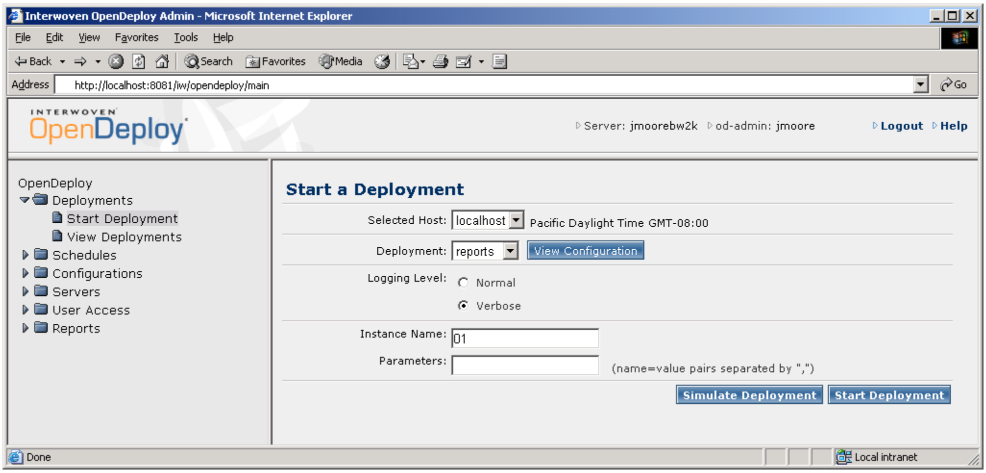
See Chapter 4, “Deployment Features” on page 171 for a complete description of all comparison-based deployability methods and criteria.

File List-Based Deployment Criteria

File list deployments do not compare files or directories, but simply deploy the files from the source host based on a specified list of files. This list of files resides on the source host and is referenced in the deployment configuration.

Graphical User Interface

OpenDeploy provides a browser-based user interface with which you can create, schedule, start, and monitor deployments. Figure 5 shows an example of the browser-based user interface. The navigation pane on the left presents you with full access to all the major task and monitoring windows. The details pane on the right displays the contents of the item in the navigation pane you selected.



Address: http://localhost:8081/iw/opendeploy/main

Server: jmoorebw2k
od-admin: jmoore

Start a Deployment

Selected Host: localhost Pacific Daylight Time GMT-08:00

Deployment: reports

Logging Level: ○ Normal ● Verbose

Instance Name: 01

Parameters: (name=value pairs separated by ",")

Figure 5: Example of the OpenDeploy User Interface

Here are some of the tasks you can perform using the OpenDeploy user interface:

- View the configuration file information for a selected deployment.
- Create a new deployment and edit the configuration of an existing one.
- Schedule deployments, either on a one-time basis, or recurrent by minute, hour, day, week, or month.
- Start a deployment.
- Monitor deployments, including logging, pending status, and success or failure.
- Generate and view reports.

Deployment Types

OpenDeploy uses one of the following methods for deploying files from the source host to the target hosts:

- Directory comparison
- File list
- TeamSite comparison

The following sections describe each of them in detail, including when each one is best suited as your deployment choice.

Directory Comparison

Directory comparison deployments compare a specified directory on the OpenDeploy server sending the files (the source host) and a corresponding server receiving the deployed files (the target host). New or updated files on the source host are then copied to the target host. After the deployment is complete, the source and target hosts' files are the same. See "File Deployment Criteria" on page 34 for information on how OpenDeploy determines which files and directories are eligible to be deployed.

You can include multiple file system locations in your deployment. The files residing in each of these specified file system locations will be compared with the target files, and the deployable files sent to the target host.

If OpenDeploy is performing a fan-out deployment, it will compare the source host's files with each of the target hosts' files. The files deployed will reflect the differences between the source host and each respective target host. That way, even if each of the target hosts have a different set of files, following the deployment, they will all be in sync with the source host and with each other.

Directory comparison deployments provide a full synchronization of the content between the source and target hosts. This is the most comprehensive way to ensure that the content in the source and target hosts are identical. Directory comparison is also the most resource-demanding method for time and network bandwidth because file and directory information must be exchanged over the network to determine the source and target hosts' file differences.

File List

A file list-based deployment does not compare files on the source and target hosts or between TeamSite areas. Instead, OpenDeploy moves files based on a predetermined list specifying the files to be deployed. The files and their paths in the list are in locations relative to the file system-based area specified in the deployment configuration. This list can be a fixed or static file, or it can be generated dynamically using some generation tool such as the TeamSite `iwevents` command-line tool. Refer to your TeamSite documentation for information on TeamSite command-line tools.

The entries in the file list are influenced by the file system syntax of the host server. Forward slashes ("/") can be used for either Windows or UNIX hosts:

```
www/index.html
www/andre/index.html
www/products.html
```

while backslashes ("\") are only permitted on Windows hosts:

```
www\index.html
www\andre\index.html
www\products.html
```

In these examples, `www` is a directory immediately subordinate to the area location on the source host as specified in the deployment configuration. For example, if the area specified for a source host is the following directory:

```
C:\webfiles
```

then the entries in the file list example combined with the specified area would end up being:

```
C:\webfiles\www\index.html
C:\webfiles\www\andre\index.html
C:\webfiles\www\products.html
```

A file list deployment is simpler and more predictable than the directory and TeamSite comparison deployments previously described. If a large number of smaller files are involved, a file list deployment can takes less of a toll on network resources than a directory comparison, and can process the deployment faster than a TeamSite comparison. However, if very large files are involved, deploying all the files specified in the file list might be less efficient than a comparison-based deployment where only those files that have changed are deployed.

TeamSite Comparison

TeamSite comparison compares two TeamSite areas within the same backing store on the source host with TeamSite installed, and deploys the differences to the target hosts. You can configure OpenDeploy to compare two TeamSite areas in any single backing store on a multi-backing store TeamSite host. The source host must have TeamSite installed and configured to use this type of deployment. The TeamSite workareas, staging areas, and editions all can be specified for a TeamSite comparison deployment. See "File Deployment Criteria" on page 34 for information on how OpenDeploy determines which files and directories are eligible to be deployed.

In a TeamSite comparison, you specify the TeamSite area containing the updated files that you want to deploy. This area is typically the branch staging area or an edition. You must also specify another TeamSite area that contains a mirror image of the files on the target hosts, typically a previously created edition. TeamSite will compare the area containing the updated files with the area containing the existing target host files and deploy the differences to the specified target hosts.

You can also simply configure OpenDeploy to determine the latest and next-to-latest TeamSite editions automatically, and deploy the differences between them. This is a common method of integrating OpenDeploy and TeamSite.

Unlike a directory comparison deployment, the TeamSite comparison takes place solely on the source host. OpenDeploy assumes that the files in the previous area are identical to those on the target hosts themselves. The deployed files are moved to the file system location on the target host specified in the deployment configuration.

Because the TeamSite comparison is done purely on the TeamSite source host, it is faster and less bandwidth-intensive than the directory comparison. No network traffic is generated before the file deployment itself occurs. However, TeamSite comparison is totally dependent on the source host having a perfect snapshot of the files residing on the target host. If files have been changed on the target host, it is up to you to ensure that the corresponding TeamSite area on the source host is updated as well. OpenDeploy can reverse-deploy from a target host to the source host, but it cannot independently determine when that will be necessary.

Deployment Scenarios

OpenDeploy allows a wide range of deployment scenarios to meet every possible need. The following sections describe the various methods with which you can deploy files between OpenDeploy hosts.

Deployment to a Single Target

The simplest deployment of files is from the source host to a single target host. In Figure 6, the OpenDeploy source host references the deployment configuration for a single target deployment. This configuration specifies a forward deployment to the target host based on one of the three types of deployments:

- Directory comparison
- TeamSite comparison
- File list

The source host subsequently deploys those files to the target host, and the deployment is completed successfully.

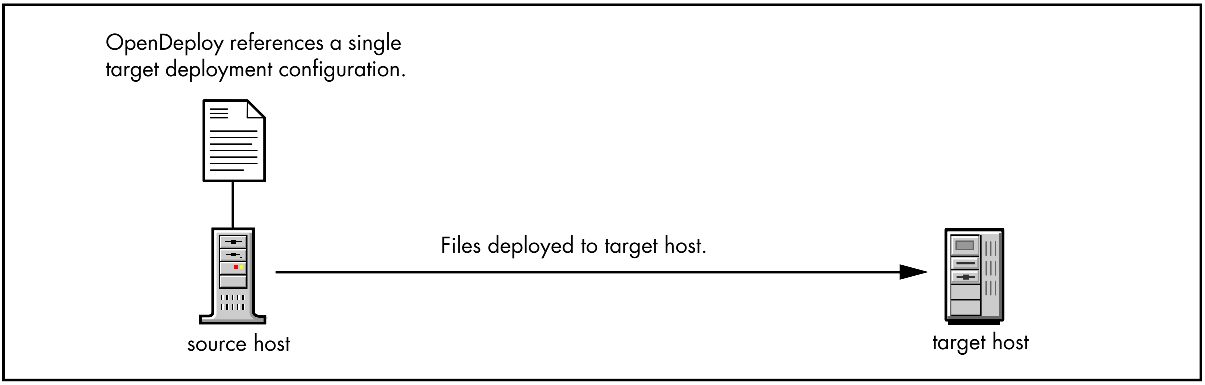

OpenDeploy references a single target deployment configuration.
Files deployed to target host.
source host
target host

Figure 6: Deployment to a Single Target Host

Deployment to Multiple Targets

This deployment configuration specifies that a set of files should be deployed to two or more target hosts. In OpenDeploy, this is referred to as a *fan-out deployment*.

In Figure 7, the OpenDeploy source host server *A* references the deployment configuration for a fan-out deployment. This configuration specifies a fan-out deployment to the target hosts *A-1*, *A-2*, and *A-3*.

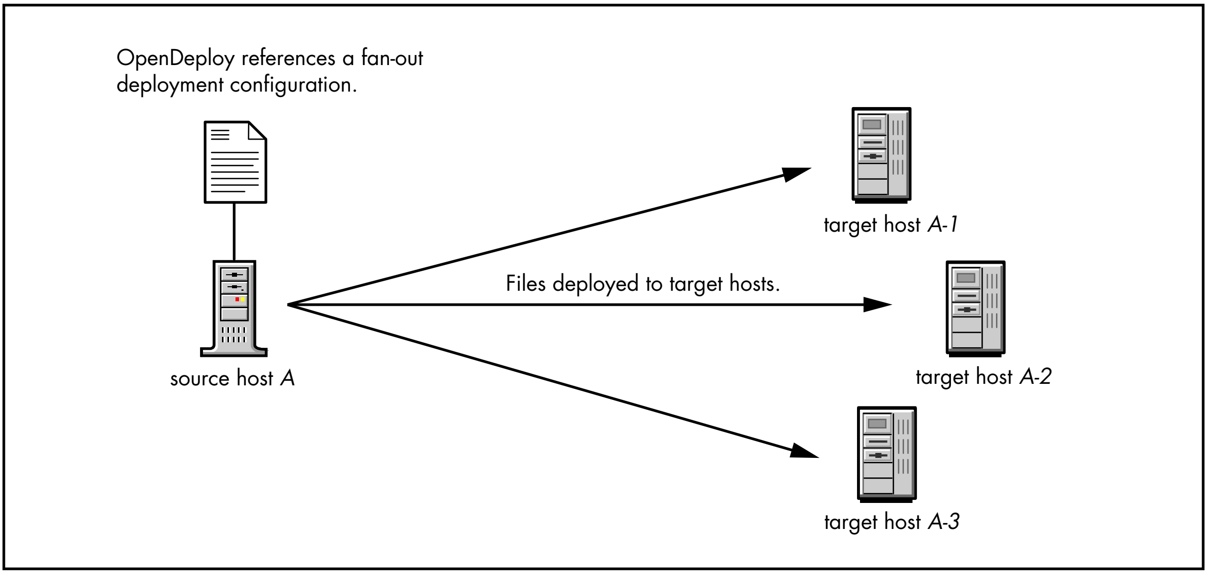

OpenDeploy references a fan-out deployment configuration.
source host *A*
Files deployed to target hosts.
target host *A-1*
target host *A-2*
target host *A-3*

Figure 7: Deployment to Multiple Target Hosts

Multi-Tiered Deployments

A *multi-tiered deployment* employs one or more target OpenDeploy servers to deploy files to another set of servers, known as a *tier*. Each source host and its target hosts represent a separate tier. Like fan-out deployments, multi-tiered deployments allow the automatic deployment of files to a wide range of recipient targets in your enterprise with little more effort than deploying to a single target host.

In Figure 8, the OpenDeploy source host *mars* performs a fan-out deployment as in Figure 7. Of the three target hosts receiving the deployed files, *venus* is an OpenDeploy host with the base server software installed and is capable of performing deployments. Like *mars*, *venus* also references its own specified deployment configuration for its own deployment of files to *mercury* and *neptune*.

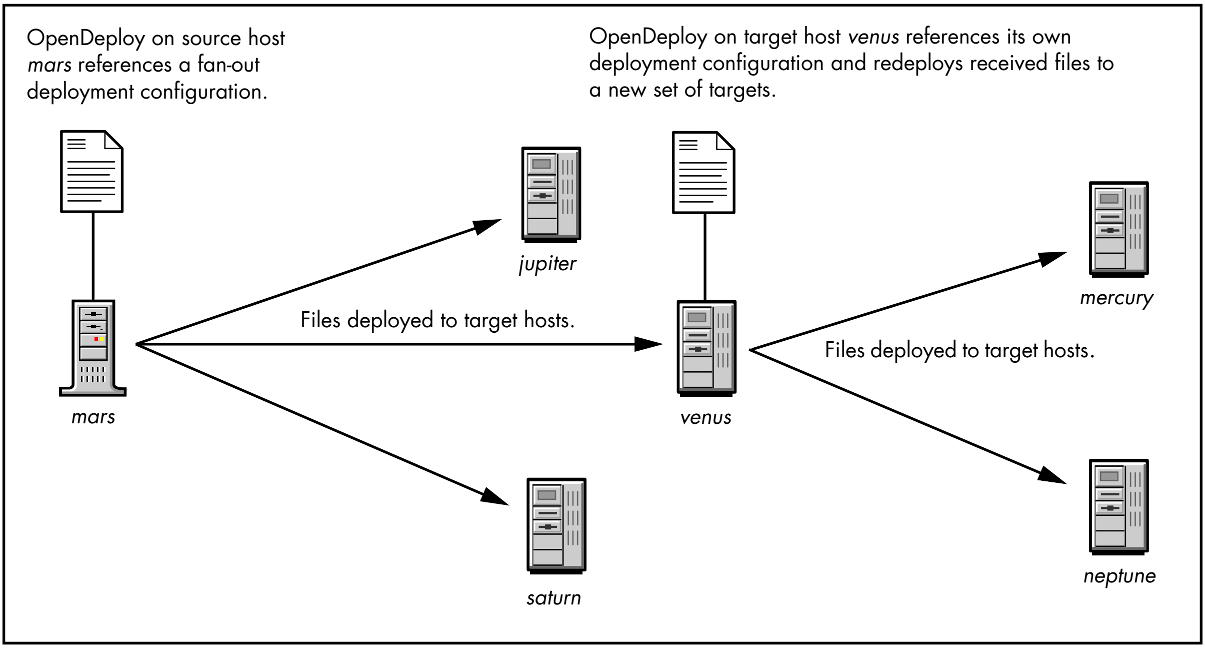

OpenDeploy on source host
mars references a fan-out
deployment configuration.
OpenDeploy on target host *venus* references its own
deployment configuration and redeploys received files to
a new set of targets.
jupiter
Files deployed to target hosts.
mercury
Files deployed to target hosts.
mars
venus
saturn
neptune

Figure 8: Multi-Tiered Deployment

Transactional Deployments

If one or more targets in a deployment fail to successfully receive the deployed files, you can configure OpenDeploy to automatically roll back the deployment that took place and restore all the target hosts to their previous states. This feature is known as a *transactional deployment*. This ability is vital if you have multiple production servers that must perfectly mirror each other. Transactional deployments ensure that disparities between servers will not exist as the result of a failed or problematic deployment. In the case of multi-tiered deployments, a transactional deployment will roll back the deployment across each tier until the entire enterprise is returned to its previous state.

In Figure 9, *mars* has run a transactional deployment to *venus* which has failed. Upon determination that the deployment failed, OpenDeploy removes the portion of the deployment that did make it to *venus*, and restores that host's files to its previous state, effecting a rollback of the failed deployment.

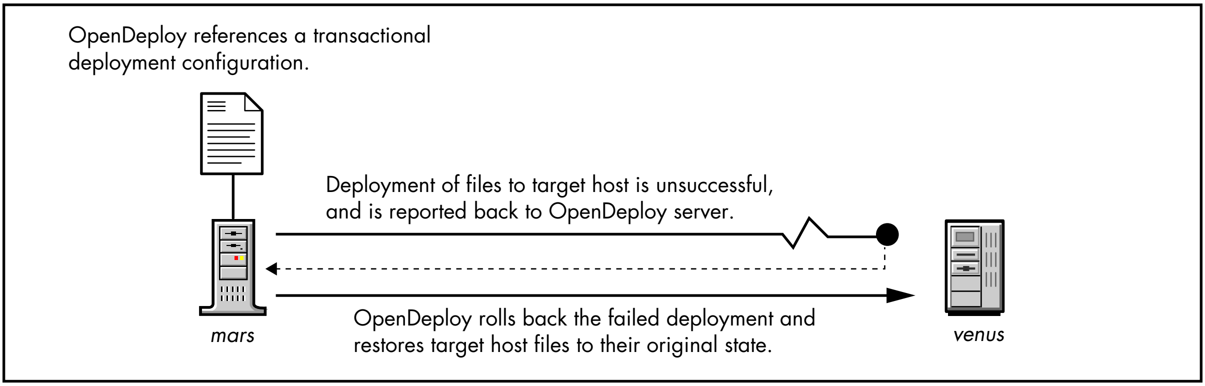

OpenDeploy references a transactional deployment configuration.
Deployment of files to target host is unsuccessful, and is reported back to OpenDeploy server.
OpenDeploy rolls back the failed deployment and restores target host files to their original state.
mars
venus

Figure 9: Transactional Deployment

Transactional deployments require temporary disk space equal to about three times the size of the deployed content. Performance time is also slower than other deployment methods.

Specifying a Target Quorum

In cases where you have multiple targets receiving a transactional deployment, it might be necessary for only a minimum number of those targets to receive their deployments for the entire deployment to be considered successful. This minimum number is called the quorum, and can be specified in the deployment configuration. Specifying a quorum prevents the entire transactional deployment from being rolled back if an acceptable number of target successfully receive their deployments.

Reverse Deployments

In some cases, the files on a production server may be changed before those on the development server. For example, production servers might generate the following types of data:

- Log files.
- Data files created via an application server.
- Assets uploaded through a Web server application.

As these production server files are generated, it might be important to deploy them back to the development server. *Reverse deployments* meet this requirement by comparing and moving files from a specified target server back to the source server.

In Figure 10, the production server has generated a number of production-related files that need to be deployed back to the development server. A reverse deployment configuration assigns the *reverse source* role to the production server, and the *reverse target* role to the development server. The deployment then takes place like any other deployment.

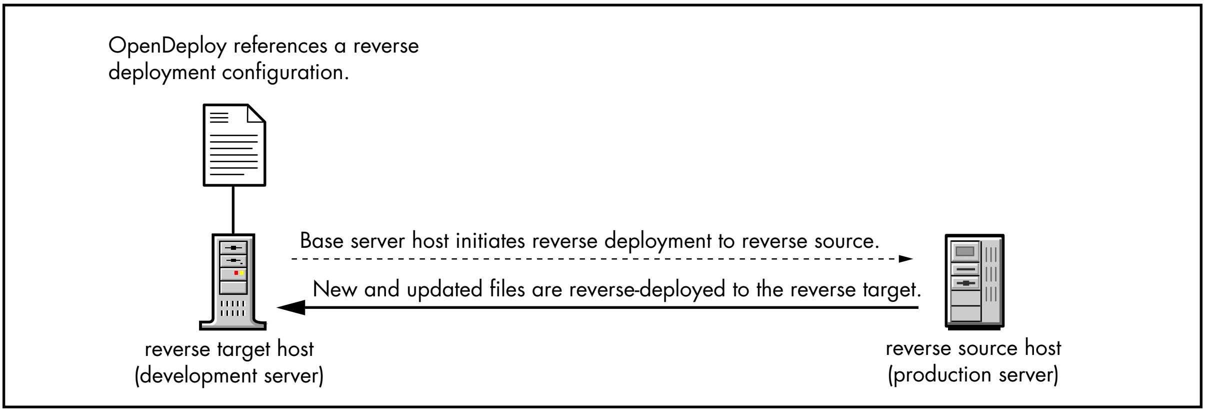

OpenDeploy references a reverse deployment configuration.
Base server host initiates reverse deployment to reverse source.
New and updated files are reverse-deployed to the reverse target.
reverse target host (development server)
reverse source host (production server)

Figure 10: Reverse Deployment

Access Rights and Privileges

In an organization with thousands of files on hundreds of servers, deployment operations need to be carefully managed and restricted only to authorized users. OpenDeploy makes a distinction between those who need the authority to create and configure deployment configurations, and those who simply need the authority to start a specific deployment. To meet this need, there are *Administrator* and *User* roles that restrict what tasks an individual can perform using the OpenDeploy user interface. These roles apply to what tasks can be performed involving specific hosts and deployments. Each individual OpenDeploy user can be assigned one or both of these roles. The individual must select which role they will be using when they login to the OpenDeploy user interface.

Administrator Role

The Administrator role allows full access to OpenDeploy configuration and functionality. The Administrator role is authorized to perform any deployment operations including the following:

- Designate which users can invoke particular deployments.
- View, edit, and upload any deployment configuration.
- Schedule, start, cancel any deployment.
- Edit existing schedules.
- Monitor status of all deployments.
- Assign Administration and User roles to the appropriate individuals.
- Generate and view reports.

User Role

The deployment User role authorizes an individual with User access to perform deployment operations on specific deployments (previously created by an individual in the Administrator role). Specifically, the User role can perform the following tasks for their deployments:

- Start and cancel deployments for which the individual is authorized.
- View the deployment configurations for which the individual is authorized.
- Create and edit schedules for which the individual is authorized.
- Monitor the status of all deployments.
- View the schedules for all deployments.
- Generate and view reports.

Chapter 2

Getting Started

This chapter introduces the basic features and functions of OpenDeploy that you can perform from the user interface and command line. For some users, this information is sufficient to meet their OpenDeploy needs. For others, particularly those who want to create more complex deployment configurations, this chapter will provide the foundation upon which they can go on to the more advanced features described later in this book.

Starting OpenDeploy

OpenDeploy is run by starting its services or daemons on the host server. This process differs depending on the type of platform on which it is operating.

Windows

Start OpenDeploy services on a Windows server by using one of the following methods:

- Rebooting
- Starting the OpenDeploy services from the Services window
- Starting from the command line

Starting by Rebooting

After the OpenDeploy software is successfully installed on the server, the OpenDeploy services will automatically start upon rebooting. Be sure to have your bootstrap administrator already configured before rebooting. Refer to "Configuring the Bootstrap Administrator" on page 51 in the *OpenDeploy Installation and Reference Guide* for more information.

Starting from the Services Window

You can start OpenDeploy services from the Services window. You might prefer this method if the OpenDeploy services were previously stopped and it is impractical to restart the server.

OpenDeploy services can vary depending on what software components you have installed. Here is a table of OpenDeploy services and their corresponding software components:

Service	Software
Interwoven OpenAPI	Operations server
Interwoven OpenDeploy UI Admin	Administration server
Interwoven OpenDeploy Service	OpenDeploy Sender (sending host only) and Receiver (receive-only host only)

To start the OpenDeploy services, follow these steps:

1. Open the Services window. This process may differ depending on which version of Windows you are using.
2. Right-click on **Interwoven OpenAPI** and select **Start** from the pop-up menu.
3. Right-click on **Interwoven OpenDeploy UI Admin** and select **Start** from the pop-up menu.
4. Right-click on the **Interwoven OpenDeploy Service** and select **Start** from the pop-up menu.

Starting From the Command Line

To start one or more OpenDeploy services from the Windows command line, follow these steps:

1. Open a command line window and navigate to the following location:

```
od-home\bin
```

2. Enter the following command to start the Interwoven OpenAPI service:

```
net start "Interwoven OpenAPI"
```

3. Enter the following command to start the Interwoven OpenDeploy UI Admin service:

```
net start "Interwoven OpenDeploy UI Admin"
```

4. Enter the following command to start the Interwoven OpenDeploy service:

```
net start "Interwoven OpenDeploy Service"
```

UNIX

Start OpenDeploy daemons on a UNIX server by using one of the following methods:

- Rebooting the server — After OpenDeploy software is successfully installed, OpenDeploy daemons will automatically start upon rebooting the server. Be sure to have your bootstrap administrator already configured before rebooting after installing OpenDeploy software on your host. Refer to “Configuring the Bootstrap Administrator” on page 51 in the *OpenDeploy Installation and Reference Guide* for more information.
- Entering the start command at the prompt — In some cases it is not practical to shut down the server to start the OpenDeploy daemons. You can start OpenDeploy without rebooting the server by navigating to the location of the OpenDeploy start program.

Starting the Operations Server

To start the operations server that was installed with OpenDeploy, follow these steps:

1. Navigate to the following location:

   `ops-home/etc/init.d` *or*

   `/etc/init.d`

2. Start the operations server by entering the following command at the prompt:

```
./opsstart start
```

Starting TeamSite OpenAPI

If you are using OpenDeploy in conjunction with TeamSite 5.5 or later, you must use the TeamSite OpenAPI instead of the OpenDeploy operations server. Refer to “Using TeamSite OpenAPI with OpenDeploy” on page 32 (Windows) and “Using TeamSite OpenAPI with OpenDeploy” on page 40 (UNIX) in the *OpenDeploy Installation and Reference Guide* for more information.

Starting the Administration Server

To start the administration server, follow these steps:

1. Navigate to the following location:

   admin-home/servletd/bin *or*

   /etc/init.d

2. Start the administration server by entering the following command at the prompt:

   ```
   ./iwtcboot start
   ```

Starting the Base Server and Receiver

To start the base server or receiver software, follow these steps:

1. Navigate to the following location:

   od-home/etc/init.d *or*

   /etc/init.d

2. Start the OpenDeploy software by entering the following command at the prompt:

   ```
   ./iwod start
   ```

Note that the `iwreset` command-line tool does not restart OpenDeploy.

Starting TeamSite and OpenDeploy on the Same Host

If you are using TeamSite and OpenDeploy on the same host, you must start them in the proper sequence:

- TeamSite
- OpenDeploy base server or receiver
- OpenDeploy administration server

You might need to start TeamSite and OpenDeploy both for the initial starting of these products, or to restart them after they have already been running. If you want to restart OpenDeploy software that had previously been running, it is important that you properly stop the software first, and then restart it. Restarting without previously stopping the software is a common source of problems.

If the OpenDeploy base server or receiver software is already running normally, you can start TeamSite without having to restart the base server or receiver.

To start TeamSite and OpenDeploy on the same host, follow these steps:

1. Navigate to the following location:

   `/etc/init.d`

2. Start TeamSite by enter the following the command at the prompt:

   ```
   ./iw.server start
   ```

   Allow TeamSite and OpenAPI to completely start up before proceeding to the next step.

3. Navigate to one of the following locations:

   `/etc/init.d/iwod` *or*

   `od-home``/etc/init.d`

4. (If OpenDeploy had previously been running) Stop the base server or receiver software by entering the following command at the prompt:

   ```
   ./iwod stop
   ```

5. Start the OpenDeploy base server or receiver software co-located on the TeamSite host by entering the following command at the prompt:

   ```
   ./iwod start
   ```

6. Ensure that the OpenDeploy base or receiver has started successfully, check the OpenDeploy server log file residing in the following location:

   od-home`/log/`*hostname*`_odbase.log` *or*

   od-home`/log/`*hostname*`_odrcvr.log`

7. Navigate to the following location:

   `/etc/init.d` *or*

   admin-home`/servletd/bin`

8. (If the administration server had previously been running) Stop the administration server by entering the following command at the prompt:

   ```
   ./iwtcboot stop
   ```

9. Start the OpenDeploy administration server software by entering the following command at the prompt:

   ```
   ./iwtcboot start
   ```

Note that the `iwreset` command-line tool does not restart OpenDeploy.

You can check the current state of the OpenDeploy process at any time by navigating to the following location:

od-home`/bin`

and entering the following command at the prompt:

```
iwodserverstatus
```

Starting the User Interface

Starting OpenDeploy does not automatically start the OpenDeploy user interface. After you have OpenDeploy running, you can access the user interface using a supported Web browser. See "OpenDeploy User Interface" on page 62 for more information.

Stopping OpenDeploy

OpenDeploy is quit by stopping its services or daemons on the host server. This process differs depending on the type of platform on which it is operating.

Windows

You must stop the various OpenDeploy services running on your Windows server to stop or quit OpenDeploy. The following services correspond to the OpenDeploy software components:

Service	Software
Interwoven OpenAPI	Operations server
Interwoven OpenDeploy UI Admin	Administration server
Interwoven OpenDeploy Service	OpenDeploy Sender (sending host only) and Receiver (receive-only host only)

To stop the OpenDeploy services, follow these steps:

1. Open the Services window. This process may differ depending on which version of Windows you are using.
2. Right-click on the **Interwoven OpenAPI** and select **Stop** from the pop-up menu to stop the operations server.

   The Interwoven OpenDeploy UI Admin service is tied to the Interwoven OpenDeploy OpenAPI service. If you stop the Interwoven OpenAPI service, you will be notified that the Interwoven OpenDeploy UI Admin service will also be stopped.

   If you are using OpenDeploy in conjunction with TeamSite, stopping this service will also stop TeamSite.
3. (If necessary) Right-click on **Interwoven OpenDeploy UI Admin** and select **Stop** from the pop-up menu to stop the administration server.
4. Right-click on the **Interwoven OpenDeploy Service** and select **Stop** from the pop-up menu to stop the base server or receiver software.

UNIX

To stop the OpenDeploy daemons on a UNIX server, follow these steps:

Stopping the Operations Server

To stop the operations server that was installed with OpenDeploy, follow these steps:

1. Navigate to the following location:

   `ops-home/iwopenapi` *or*

   `/etc/init.d`

2. Stop the operations server by entering the following command at the prompt:

   ```
   ./opsstart stop
   ```

Stopping TeamSite OpenAPI

Refer to your TeamSite documentation for instructions on how to stop TeamSite OpenAPI.

Stopping the Administration Server

To stop the administration server, follow these steps:

1. Navigate to the following location:

   `admin-home/servletd/bin` *or*

   `/etc/init.d`

2. Stop the administration server by entering the following command at the prompt:

   ```
   ./iwtcboot stop
   ```

Stopping the Base Server and Receiver

To stop the base server or receiver, follow these steps:

1. Navigate to the following location:

   `od-home/etc/init.d` *or*

   `/etc/init.d`

2. Stop the base server or receiver software by entering the following command at the prompt:

   ```
   ./iwod stop
   ```

Running OpenDeploy as Non-Administrator or Non-Root

You can configure OpenDeploy to run as someone other than the Administrator user on Windows or the root user on UNIX. The method differs depending on whether OpenDeploy is running on Windows or UNIX.

Running OpenDeploy on Windows as Non-Administrator

You can run OpenDeploy on Windows as non-Administrator by reconfiguring the Windows settings.

To run OpenDeploy on Windows as non-Administrator, follow these steps:

1. Open the Services window. This process may differ depending on the version of Windows you are using.
2. Stop the **Interwoven OpenDeploy Service** in the Services window.

   See "Stopping OpenDeploy" on page 53 for more information on stopping OpenDeploy services.
3. Perform the following task depending on which Windows operating system you are using:
   - Windows 2000: select **Action** > **Properties** to open the Properties window. You must select any item in the **Name** column of the Services window for this command to be available. Next, select the **Log On** tab to make it active.
   - Windows NT: click **Startup** to open the Service window.
4. Click **This account** and enter the account user and password information in the appropriate boxes.

5. Click OK and close the Services window.
6. Restart the OpenDeploy service you had previously stopped. See "Starting OpenDeploy" on page 47 for more information.

Running OpenDeploy on UNIX as Non-Root

You can convert an OpenDeploy base server or receiver host running on UNIX to run as non-root using the `iwodnonroot` command-line tool. The `iwodnonroot` command changes the ownership on the required OpenDeploy directories and files from root to a specified user and group ID. When OpenDeploy is run as that user and group ID, it can write and access the necessary files.

You must be root to install OpenDeploy, and subsequently to run the `iwodnonroot` command.

Here is a listing of the `iwodnonroot` usage syntax:

`iwodnonroot` *`owner group`* [*`od-home`*]

`-h`	Displays help information.
`owner`	User name or ID
`group`	Group name or ID
`od-home`	Full path to the OpenDeploy home directory.

If the optional *`od-home`* argument is not specified, `iwodnonroot` will lookup the *`od-home`* from the following file:

```
/etc/defaultiwodhome
```

If *`od-home`* is specified, `iwodnonroot` will apply the `chown/chgrp` changes to the specified *`od-home`* location.

To run OpenDeploy on UNIX as non-root, follow these steps:

1. Stop the OpenDeploy base server or receiver daemon.

   See "Stopping OpenDeploy" on page 53 for more information on stopping OpenDeploy daemons.

2. Navigate to the following directory:

   ```
   od-home/bin
   ```

3. Start the conversion by entering the following command at the prompt:

   ```
   iwodnonroot owner group
   ```

   For example, if you wanted to convert OpenDeploy to run under the user ID *jdoe* and group ID *tech-pubs*, you would enter:

   ```
   iwodnonroot jdoe tech_pubs
   ```

   If you wanted to include the `od-home` location, the command might be:

   ```
   iwodnonroot jdoe tech_pubs /usr/local/OpenDeployNG
   ```

4. Restart, as a user other than root, the OpenDeploy daemon you had previously stopped. See "Starting OpenDeploy" on page 47 for more information on starting OpenDeploy daemons.

Refreshing the OpenDeploy Server

Any time you modify certain elements in the base server, receiver, or nodes configuration files, you must reset your OpenDeploy base server or receiver service or daemon to accept the changes. You can refresh your OpenDeploy server without rebooting or manually restarting individual services or daemons by using the `iwodserverreset` command-line tool. The `iwodserverreset` command-line tool refreshes the OpenDeploy server with the new settings specified in the updated configuration files. Using this tool, you can make changes to the server configuration file and have it be read without having to bring down your services or restart your host.

The `iwodserverreset` command-line tool will not cause the configuration to be refreshed if there are deployments in progress at the time the command is run.

The following elements in the base server and receiver configuration files (by default `odbase.xml` and `odrcvr.xml`) can be refreshed using `iwodserverreset`:

- `allowedHosts`

- `allowedDirectories`
- `logRules`

The following elements and their attributes in base server and receiver configuration file are not refreshable:

- `localNode`
- `initiatorProperties`
- `listenerProperties`
- `transportProperties`
- `schedulerProperties`

To make the server read and implement changes in these elements, you must restart your OpenDeploy services or daemons.

Refer to the *OpenDeploy Installation and Reference Guide* for descriptions of these elements and their attributes.

To refresh your OpenDeploy server's configurations, follow these steps:

1. Navigate to the following directory:

   od-home/bin

2. Reset the OpenDeploy services by entering the following command at the prompt:

   iwodserverreset

There are various options associated with the `iwodserverreset` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodserverreset -h | -v | -q
```

`-h`	Displays usage information.
`-v`	Displays version information.

`-q`	Disables messages generated when there are active deployments in progress at the time `iwodserverreset` is run.

Dependencies on the Operations Server

One of the underlying services of Interwoven products such as OpenDeploy and TeamSite 5.0 and later is OpenAPI. The operations server that comes with OpenDeploy is a subset of the OpenAPI. If you would like more information about OpenAPI, consult the administration chapter of the *OpenAPI Developer's Guide*, available on-line as part of the OpenAPI SDK.

Administration Server

The operations server or OpenAPI server must be up and running before the user interface administration service can be started.

Verifying the Operations Server

To verify that the operations server has been successfully started after an installation, follow these steps

Windows

The Services window should display "Interwoven OpenAPI," running as "administrator," with status "started." If the OpenAPI service is not running, then it needs to be restarted.

UNIX

On UNIX, the following command should display the OpenAPI daemon running:

```
ps -ef | grep IWServiceServer
```

If the OpenAPI daemon is not running, then it needs to be started manually.

Verifying OpenAPI Operation

Verifying OpenAPI operation varies depending on which release of TeamSite you are using in conjunction with OpenDeploy.

Pre-TeamSite 5.5.1

Supported releases of TeamSite OpenAPI prior to 5.5.1 can be verified in the same manner as for the operations server listed in the previous section.

TeamSite 5.5.1 and Later

TeamSite 5.5.1 and later releases have consolidated the OpenAPI server into the TeamSite administration server software. There is no longer a separate OpenAPI service on the Windows platforms, nor a separate OpenAPI server daemon on the UNIX platforms. If OpenDeploy resides on a host with TeamSite 5.5.1 or above, restarting the TeamSite administration server will also restart the TeamSite OpenAPI. There is no longer a need to restart these two servers separately. The TeamSite `iwreset` command will also restart the TeamSite administration server.

RMI Registry Service Considerations

Both the operations server, including OpenAPI, and the base server make use of the RMI registry service. Interwoven products using this service can work together without a problem. However, other software products that make use of the RMI registry can cause conflicts that might prevent OpenDeploy and other Interwoven products from working properly. It is strongly recommended that no other programs accessing the same RMI registry as the ones used by OpenDeploy and TeamSite be included in your OpenDeploy environment.

The RMI registry ports being used are:

- Operations server (including OpenAPI) — `1099`
- OpenDeploy — `9173` (default) or a port you specify during installation.

Installing TeamSite in an OpenDeploy Environment

If you are installing TeamSite 5.0.1 or later into an environment where OpenDeploy already exists, you must uninstall the OpenDeploy operations server software prior to performing the TeamSite installation. Included in the TeamSite installation is OpenAPI. OpenDeploy can use OpenAPI in place of the operations server for access and role management. After installing TeamSite, you must reconfigure OpenDeploy to use OpenAPI. Refer to "Using TeamSite OpenAPI with OpenDeploy" on page 32 (Windows) and "Using TeamSite OpenAPI with OpenDeploy" on page 40 (for UNIX) in the *OpenDeploy Installation and Reference Guide* for more information.

Adding Roles to the Operations Server

By default, the operations server software is installed with its `od-admin` and `od-user` roles. However, in some cases, these roles can be damaged or missing. If this occurs, it might be necessary to manually install them using the `iwuser` command-line tool. The implementation for using `iwuser` to add roles to an OpenDeploy operations server is different than for adding roles to TeamSite OpenAPI, as described in "Using TeamSite OpenAPI with OpenDeploy" on page 32 (Windows) and "Using TeamSite OpenAPI with OpenDeploy" on page 40 (for UNIX) in the *OpenDeploy Installation and Reference Guide*. Do not use that implementation to add roles to the OpenDeploy operations server. Instead, use the following instructions.

To add roles to the OpenDeploy operations server, follow these steps:

1. Navigate to the following location:

   ```
   ops-home/iwopenapi
   ```

2. Enter the following commands at the prompt:

   ```
   iwuser -c -role od-admin -D -F ops-home/iwopenapi/entities
   ```

   ```
   iwuser -c -role od-user -D -F ops-home/iwopenapi/entities
   ```

   Each command will run and will give the following error:

   ```
   ERROR: TeamSite's install directory (c:\iw\iw-home) does not exist, please
   make sure TeamSite is installed correctly.
   ```

   This error does not effect the `iwuser` command from functioning correctly and can be ignored.

OpenDeploy User Interface

The user interface enhances your ability to perform and monitor OpenDeploy tasks anywhere in the enterprise. All that is needed is a supported Web browser, which avoids the need to install additional client software on each computer from which you might want to administer the product. Refer to the *OpenDeploy Release Notes* for a list of supported browsers. You must also have the OpenDeploy administration server software or an alternate Tomcat server installed and its service or daemon running.

The user interface is a tool to help you learn the product and start using OpenDeploy right away. You will also find the graphical monitoring and scheduling features an advantage in managing deployments. However, more advanced features and configurations will still require you to work directly with configuration files and command-line tools.

If you are starting OpenDeploy for the first time after installing it, you must follow the procedures for first time start-up and login. After you have configured your server to recognize specific hosts, and have created the Administrator and User roles for individuals, you can follow the normal procedures for starting the OpenDeploy user interface.

Browser Refresh Requirements

To ensure that changes you make to OpenDeploy are reflected in the user interface, you should configure your Web browser to refresh a page each time it is visited.

Accessing the User Interface

To access the OpenDeploy user interface, point your browser to the following URL:

```
http://admin-server-hostname:admin-port-number/iw/opendeploy/login
```

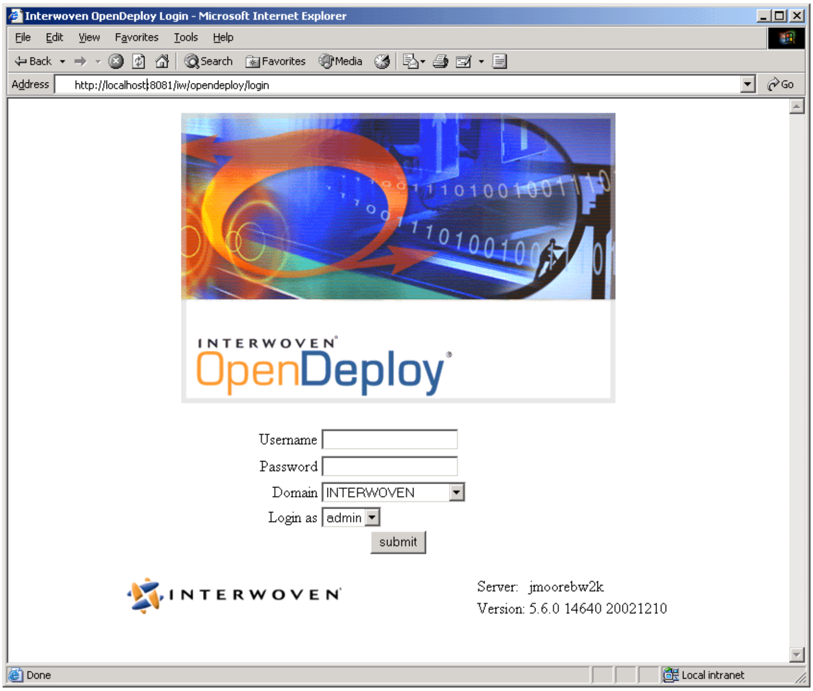
where `admin-server-hostname` is the name of the host server running the administration server software, and `admin-port-number` is the administration server port number you chose when you installed the administration server software. For example, if your administration server host was named *andromeda* and you entered the administration port number `8081`, then the URL to access the user interface would be:

```
http://andromeda:8081/iw/opendeploy/login
```

If you access the user interface from the same server on which the administration server software is running, you can use “localhost” as the administration server in the URL.

Login

Starting the user interface displays the login window (Figure 11).



Address: http://localhost:8081/iw/opendeploy/login

Username
Password
Domain INTERWOVEN
Login as admin

Server: jmoorebw2k
Version: 5.6.0 14640 20021210

Figure 11: OpenDeploy Login Window on Windows Server

The standard method of logging into OpenDeploy requires entering the following information:

- User name
- Password
- Domain (Windows only)
- Role — either `admin` or `user`. See “Roles and Authorization” on page 75 for more information.

First Time Login Using Bootstrap

When OpenDeploy is first installed on a server, you must log in as the bootstrap administrator to gain full administrator access. If you have not configured a bootstrap administrator yet, you must do so before you can log in. Refer to "Configuring the Bootstrap Administrator" on page 51 in the *OpenDeploy Installation and Reference Guide* for more information. After you have logged on as the bootstrap administrator, you need to verify and add your OpenDeploy server host to the list of OpenDeploy hosts that are accessible by the administration server. See "Adding OpenDeploy Hosts" on page 65 for more information.

Timeout Setting

OpenDeploy includes a timeout feature for the user interface. This timeout feature is a security measure to prevent unauthorized access to an OpenDeploy user interface window that has been idle past the timeout period. Users who attempt to perform any task through the user interface past the timeout period will have to log in again. The default timeout period is measured in minutes. The default period is 9999 minutes. However, you can change this value to any amount you want.

To change the timeout value of the OpenDeploy user interface, follow these steps:

1. Open the following file using your favorite text or XML editor:

   ```
   admin-home/httpd/iwwebapps/opendeploy/WEB-INF/web.xml
   ```

2. Enter the amount of time in minutes that you want for the timeout value in the following location within the `web.xml` file:

   ```
   <session-config>
      <session-timeout>xxxx</session-timeout>
   </session-config>
   ```

   where *`xxxx`* is the number of minutes you want. If you are changing this value for the first time, the existing value will be `9999`.

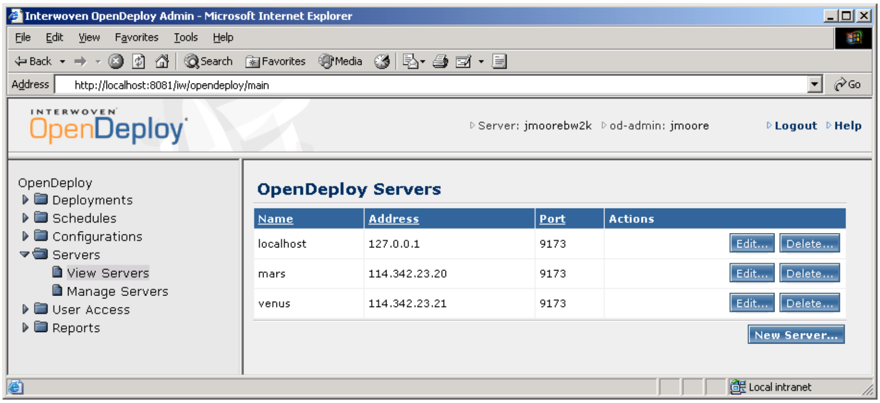
3. Save your changes and close the `web.xml` file.

4. Restart your administration server service or daemon. See "Starting OpenDeploy" on page 47 for more information.

The OpenDeploy user interface automatically will timeout when idle past the number of minutes you entered.

OpenDeploy Host Management

You can manage multiple OpenDeploy hosts from a single browser running the OpenDeploy user interface. You can apply OpenDeploy features and functionality available through the user interface to any OpenDeploy host listed, assuming you have the proper access and permissions, and the OpenDeploy host is capable of performing the particular task. You must add each OpenDeploy host you want to manage into the OpenDeploy Servers window (Figure 12). You can also modify and delete existing servers from this window.



OpenDeploy Servers

Name	Address	Port	Actions
localhost	127.0.0.1	9173	Edit... Delete...
mars	114.342.23.20	9173	Edit... Delete...
venus	114.342.23.21	9173	Edit... Delete...

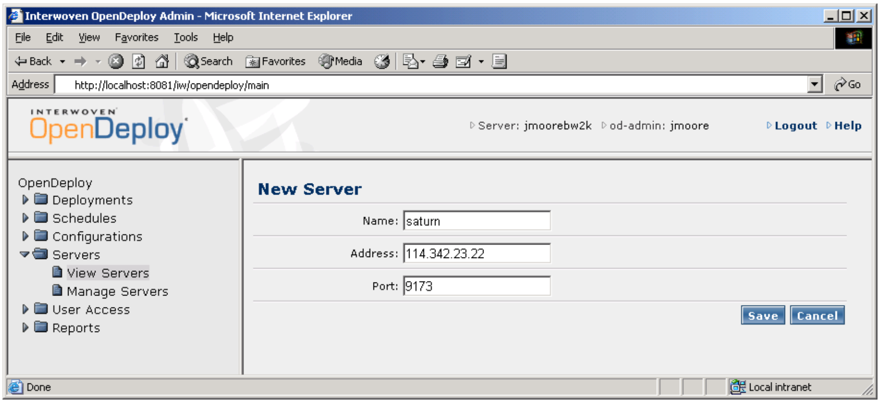
Figure 12: OpenDeploy Servers Window

Adding OpenDeploy Hosts

To add an OpenDeploy host to your user interface, follow these steps:

1. Select **Servers > View Servers** to display the OpenDeploy Servers window. Here you can see which OpenDeploy hosts are currently listed, including their fully qualified DNS host name or IP address, and RMI registry port number.

2. Click **New Server** to display the New Server window (Figure 13).



New Server

Name: saturn

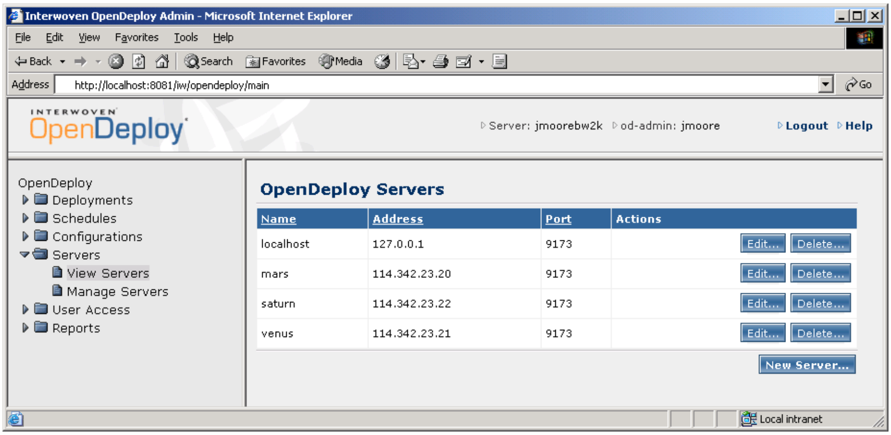
Address: 114.342.23.22

Port: 9173

Figure 13: New Server Window

3. Input the following information regarding the new host you are adding into the new host template:
   - **Name** — the logical name you define for the host.
   - **Address** — the fully qualified DNS host name or IP address of the host.
   - **Port** — the port assigned to the RMI registry service.

4. Click **Save** to add the new server. The OpenDeploy Servers window reappears (Figure 14) with the added host.



OpenDeploy Servers

Name	Address	Port	Actions
localhost	127.0.0.1	9173	Edit... Delete...
mars	114.342.23.20	9173	Edit... Delete...
saturn	114.342.23.22	9173	Edit... Delete...
venus	114.342.23.21	9173	Edit... Delete...

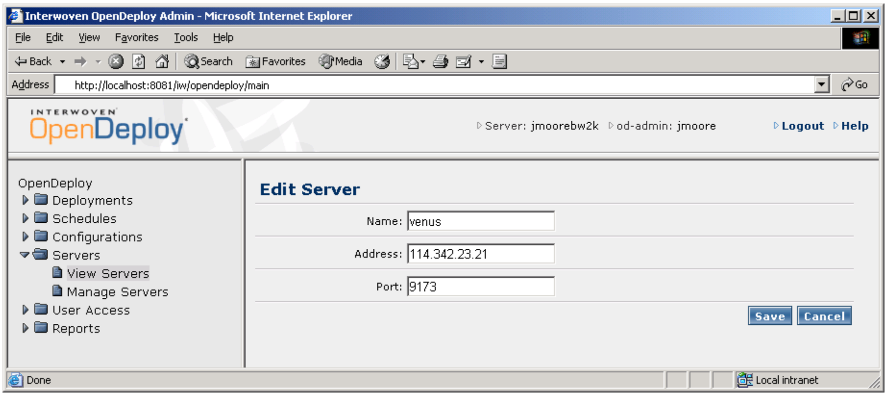
Figure 14: OpenDeploy Servers Window with Added Server

Changing Server Information

You can change the name, IP address, and RMI registry service port for any OpenDeploy host listed in the OpenDeploy Servers window.

To change the information of a selected OpenDeploy host, follow these steps:

1. Select **Servers > View Servers** to display the OpenDeploy Servers window.
2. Click the **Edit** button that corresponds to the OpenDeploy host server whose information you want to modify. The Edit Server window (Figure 15) appears.



Address http://localhost:8081/iw/opendeploy/main

Server: jmoorebw2k od-admin: jmoore

Edit Server

Name: venus

Address: 114.342.23.21

Port: 9173

Figure 15: Edit Server Window

3. Make your modifications as necessary and click **Save**. The OpenDeploy Servers window reappears, reflecting the changes you made to the host.

Deleting OpenDeploy Hosts

You delete hosts from the OpenDeploy user interface through the OpenDeploy Servers window. You are only deleting the listing of that server in the user interface, not deleting any OpenDeploy software from the actual server. You can add any server you have previously deleted at any time.

To delete a selected OpenDeploy host from the user interface, follow these steps:

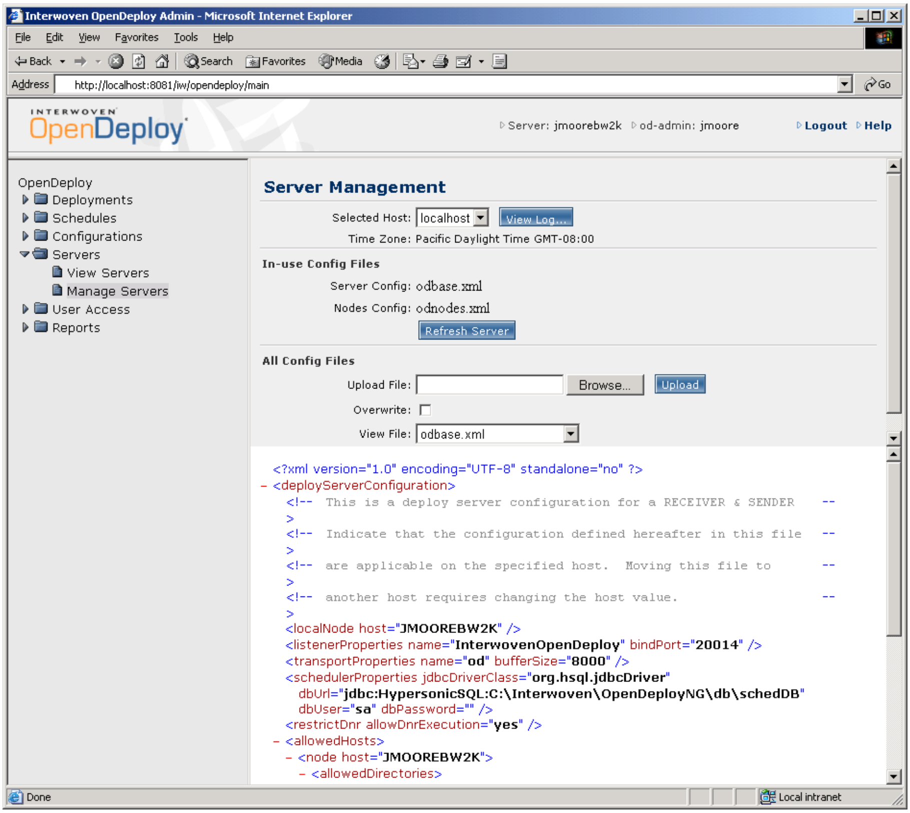
1. Select **Servers > View Servers** to display the OpenDeploy Servers window (Figure 12).

2. Click the **Delete** button that corresponds to the OpenDeploy host server that you want to delete from the user interface.

3. Click **OK** when prompted to confirm that you want to delete that selected host. The OpenDeploy Servers window is refreshed, no longer displaying the server you just deleted.

Monitoring Host Logs and Configurations

The Server Management window (Figure 16) provides a single location in the user interface where you can monitor and access the configuration and log files associated with a selected OpenDeploy host.



Server Management

Selected Host: localhost

Time Zone: Pacific Daylight Time GMT-08:00

In-use Config Files

Server Config: odbase.xml

Nodes Config: odnodes.xml

All Config Files

Upload File:

Overwrite: □

View File: odbase.xml

```
  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <deployServerConfiguration>
    <!--  This is a deploy server configuration for a RECEIVER & SENDER   --
    >
    <!--  Indicate that the configuration defined hereafter in this file  --
    >
    <!--  are applicable on the specified host.  Moving this file to      --
    >
    <!--  another host requires changing the host value.                  --
    >
    <localNode host="JMOOREBW2K" />
    <listenerProperties name="InterwovenOpenDeploy" bindPort="20014" />
    <transportProperties name="od" bufferSize="8000" />
    <schedulerProperties jdbcDriverClass="org.hsql.jdbcDriver"
      dbUrl="jdbc:HypersonicSQL:C:\Interwoven\OpenDeployNG\db\schedDB"
      dbUser="sa" dbPassword="" />
    <restrictDnr allowDnrExecution="yes" />
  - <allowedHosts>
    - <node host="JMOOREBW2K">
      - <allowedDirectories>
```

Figure 16: Server Management Window

Within the Server Management window you can perform the following host configuration and log file tasks:

- View the OpenDeploy base server and receiver log files of the selected OpenDeploy host.
- View the names of the host configuration files currently in use by the selected OpenDeploy host.
- Upload a host configuration file that you created or modified locally to a selected OpenDeploy host.
- Refresh the OpenDeploy host services and daemons to reread and implement changes in configuration.
- Display the XML code of a host configuration file located in the `*od-home*/etc` directory of a selected host.

Viewing the Base Server and Receiver Log Files

You can access and view the base server and receiver log files from the Server Management window. Refer to the *OpenDeploy Administration Guide* for information on the contents of the base server and receiver log files.

To view the base server and receiver log files, follow these steps:

1. Select **Servers > Manage Servers** to display the Server Management window.
2. Select the name of the OpenDeploy server whose base server or receiver log you want to view from the **Selected Host** list.
3. Click **View Log**. A separate browser window appears displaying the OpenDeploy LogViewer window containing the base server or receiver log file of the host you chose.

Uploading Host Configuration Files

The Server Management window provides you the ability to upload host configuration files (base server, receiver, and nodes configuration files) from your local computer into the *od-home*/etc directory of the selected host. Any file you upload must be a valid XML file that follows the deploy server or nodes configuration DTD.

For example, if you wanted to add or modify the base server file of a particular OpenDeploy host, you could create or make the appropriate changes to that file and upload it to that host. You must still use a text or XML editor to modify the configuration file. You cannot modify the file from the Server Management window, only copy it to the host. This feature is only available to individuals holding an Administrator role on the affected OpenDeploy host. See “Roles and Authorization” on page 75 for more information.

Any modified configuration file you upload must have the same file name as the one you intend to replace. Names of the configuration files currently in use by the selected OpenDeploy host are displayed in the **In-use Config Files** section of the Server Management window. You can also upload other files with the .xml extension, but the OpenDeploy host will not be able to read those files upon refreshing. Additional steps are required to substitute a differently named host configuration file for one currently in use by an OpenDeploy host.

If you want to modify a host configuration file currently in use by the selected OpenDeploy host, you are responsible for obtaining a copy of that file and placing it on your local computer. You cannot download host configuration files through the OpenDeploy user interface. You will have to map a drive to the selected host or to find another method to copy the file you want from the selected host to your local computer.

To upload a host configuration file to a remote OpenDeploy host, follow these steps:

1. Create a new host configuration file or modify an existing one on your local computer.
2. Select **Servers > Manage Servers** to display the Server Management window.
3. Select the name of the OpenDeploy server to which you want to upload files from the **Selected Host** list.

4. Enter the path to the configuration file you want to upload to the selected host in the **Upload File** box. You can also click **Browse** and navigate to the location of the file. Any file you upload must have the `.xml` extension.

5. Check the **Overwrite** box. This is required any time you are overwriting an existing host file.

6. Click **Upload** to copy the file you specified from your computer to the selected OpenDeploy host. Your file will be copied to the *`od-home`*`/etc` directory, and will overwrite the existing file there.

The OpenDeploy host receiving your uploaded file will not run with the changes in the updated configuration file until you refresh the server.

Viewing Host Configuration Files

You can view the XML source code of the host configuration files, and other XML-based files that reside in the *`od-home`*`/etc` directory of a selected OpenDeploy host, through the Server Management window. This is a quick and convenient way to see how a selected OpenDeploy host is configured. Only valid XML files will be displayed. Any other file will produce an error message, even if it appears in the **View File** list. This feature is only available to individuals holding an Administrator role on the selected OpenDeploy host. See “Roles and Authorization” on page 75 for more information.

To view the source code of an OpenDeploy host configuration file, follow these steps:

1. Select **Servers > Manage Servers** to display the Server Management window.

2. Select the name of the OpenDeploy server whose host configuration file source code you want to view from the **Selected Host** list.

3. Select the file whose source code you want to view from the **View File** list. The source code is displayed in the **Configuration** window.

If a file you want to view does not appear in the **View File** list, ensure that the file resides in the *`od-home`*`/etc` directory of the selected host. If you receive an error message after selecting a file, it may not be a properly formatted XML file.

Refreshing the OpenDeploy Host Server

You can refresh the selected OpenDeploy host through the Server Management window. Refreshing the host causes it to reread its host configuration files currently in use. These files are displayed in the Server Management window (Figure 16) under **In-use Config Files**. Any changes you made to certain elements in the base server, receiver, or nodes configuration files in use will only be read and followed when you refresh that host. This applies both to changes you make to host configuration files that you uploaded through the Server Management window, and to changes you make by modifying the configuration files directly on the host. The refresh feature in the Server Management window functions the same as the `iwodserverreset` command-line tool. See "Refreshing the OpenDeploy Server" on page 57 for more information, including a list of those supported elements.

For example, if you modify the log file directory of the base server configuration file of a selected host, the new log file directory will only be used after the host is refreshed. Any configuration change you make will only be applied to actions that occur after the host is refreshed. OpenDeploy will not retroactively apply changes, such as moving the older log files from their previous location to the new one you specified.

Refreshing OpenDeploy only applies to the host configuration files whose names appear in the **In-use Config Files** section of the Server Management window. OpenDeploy hosts with the base server software installed will display the base server configuration file name (default `odbase.xml`) and the nodes configuration file (default `odnodes.xml`), while hosts with the receiver software installed will only display the name of the receiver configuration file (default `odrcvr.xml`).

You cannot change the name of host configuration files in use by a selected host through the OpenDeploy user interface, nor can you select a different configuration file. Changing the host configuration files can only be done by modifying the host service configuration file (`deploy.cfg`) and restarting the OpenDeploy services or daemons. Refer to "Modifying the Service Configuration File" on page 47 in the *OpenDeploy Installation and Reference Guide* for more information about modifying the base server service and receiver service configuration files.

As the refresh on the selected host takes place, the progress is logged in the base server or receiver log file. Check these logs to determine when the refresh procedure has completed before resuming OpenDeploy tasks.

This feature is only available to individuals holding an Administrator role on the affected OpenDeploy host. See "Roles and Authorization" on page 75 for more information.

To refresh a selected OpenDeploy host's configuration files, follow these steps:

1. Select **Servers > Manage Servers** to display the Server Management window.
2. Select the name of the OpenDeploy server whose host configuration files you want to refresh from the **Selected Host** list.
3. Click **Refresh Server** to begin the refreshing procedure.
4. Click **View Log** to display a separate browser window containing the OpenDeploy Log Viewer window. Here you can view the base server or receiver log file to follow the progress of the refresh.

Determining the OpenDeploy Server Version

You can determine which version of OpenDeploy you are running by using the `iwodservergetversion` command-line tool.

To determine the version of your OpenDeploy server, follow these steps:

1. Navigate to the following directory:

   ```
   od-home/bin
   ```

2. Display the version by entering the following command at the prompt:

   ```
   iwodservergetversion
   ```

The `-h` option associated with the `iwodservergetversion` command-line tool will display help information.

```
iwodservergetversion -h
```

`-h`	Displays help information.

Displaying the OpenDeploy Server Status

You can display the status of the OpenDeploy server, including its registry port and the number of active deployments, by using the `iwodserverstatus` command-line tool.

To display the status of your OpenDeploy server, follow these steps:

1. Navigate to the following directory:

   `od-home``/bin`

2. Display the status by entering the following command at the prompt:

   `iwodserverstatus`

There are various options associated with the `iwodserverstatus` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodserverstatus [-h | -v | -q]
```

`-h`	Displays help information.
`-v`	Displays version information.
`-q`	Omits displaying the number of active deployments.

Roles and Authorization

OpenDeploy recognizes two levels, or *roles*, of access to the product's features and functionality:

- Administrator
- User

The role an individual has determines what tasks that person can perform within the OpenDeploy environment, and on specific OpenDeploy hosts.

Administrator Role

The *Administrator role* allows full access to OpenDeploy configuration and functionality. The Administrator role is authorized to perform the following tasks:

- Scheduling, starting, and cancelling all deployments.
- Monitoring status of all deployments.
- Viewing all schedules for deployments.
- Viewing the XML source code of a deployment configuration.
- Importing a deployment configuration file into OpenDeploy.
- View the OpenDeploy server log.
- View and upload OpenDeploy server configuration files.
- Create deployment configurations.
- Adding additional individuals to Administrator and User roles.
- Designating which individuals holding User roles can invoke particular deployments.

User Role

The *User role* authorizes an individual with User access to perform deployment operations on specific deployments (previously created by an individual with an Administrator role). Specifically, the User role can perform the following tasks for their assigned deployments:

- Scheduling, starting, and cancelling deployments.
- View the schedules for the deployments.
- Monitoring status.
- Viewing the XML source code of a deployment configuration.
- View the OpenDeploy server log.

For example, both *User1* and *User2* belong to the same Web Operation user group with access to Web server *mars* at the operating system level. *User1* is authorized to manage the deployment for the Residential Mortgage Web site, while *User2* is authorized to manage the deployments for the Brokerage Web site. Both *User1* and *User2* have access to the same development server, but each is allowed to launch only the deployment assigned to them.

Managing Role Access

Individuals with an Administrator role can assign or revoke Administrator and User roles to others. Role management includes performing the following tasks for each individual:

- Specifying the system role access, which modifies the operations server role database. This determines what role an individual can hold within the OpenDeploy environment as a whole.
- Specifying the server role access, which modifies the OpenDeploy server role database of the particular host server. This determines what role an individual can hold on a specific OpenDeploy host.

The system role access must be specified for an individual before you can specify their server role access.

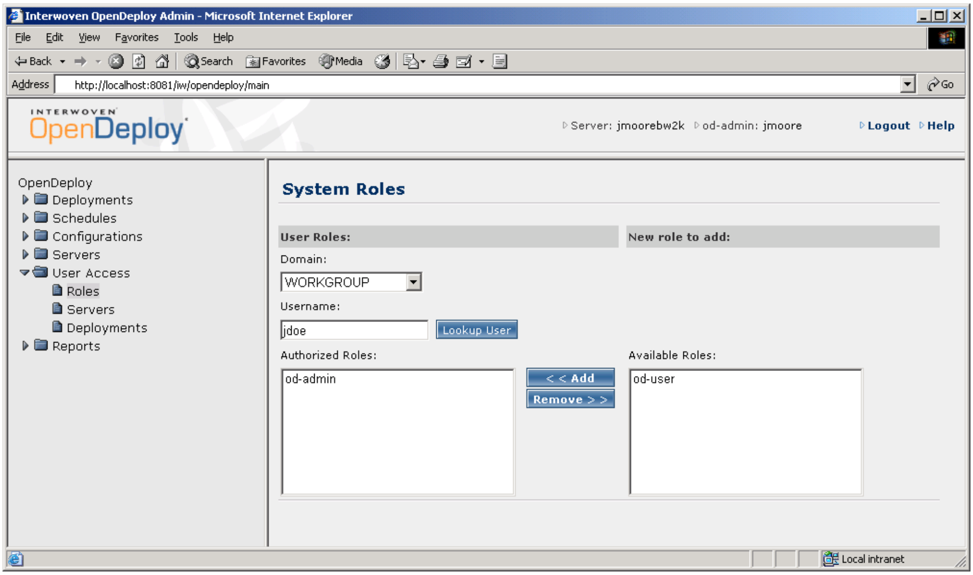
Specifying System Role Access

Specifying the system role access of an individual determines under what role that person can log in to the OpenDeploy user interface. The operations server or OpenAPI software maintains the role database for system roles. The role under which the individual logs in to the OpenDeploy environment determines in part what subsequent role that individual can have on a server-by-server basis. Here is a description of each system access role:

- **Admin** — logging in as `admin` gives the following access:
  - Access to all hosts in the OpenDeploy environment.
  - Ability to perform administrator tasks on the OpenDeploy hosts where the individual has been assigned the Administrator role.
- **User** — logging in as `user` gives the following access:
  - Read-only access to all hosts in the OpenDeploy environment.
  - Ability to perform user tasks on the OpenDeploy hosts where the individual has been assigned the User role.

An individual must be assigned a system role prior to being assigned an Administrator or User role on a specific OpenDeploy host.

You can assign and manage system role access through the user interface in the System Roles window (Figure 17).



Address: http://localhost:8081/iw/opendeploy/main

Server: jmoorebw2k
od-admin: jmoore

System Roles

User Roles:
New role to add:

Domain: WORKGROUP

Username: jdoe

Authorized Roles: od-admin

Available Roles: od-user

Figure 17: System RolesWindow on a Windows Host

The System Roles window allows you to specify an individual in your enterprise and assign that person a system role. All individuals must have an existing user account on the operations server host to be assigned a system role.

Individuals new to the OpenDeploy system role environment will initially have no authorized role established. Individuals who already have a system role assigned will have that role displayed in the **Authorized Roles** list. As the administrator, you must determine the needs of individuals you add, and assign them the appropriate role. If the individual needs administrative (od-admin) access on even one OpenDeploy host, then that individual must be given administrator login access. Otherwise, you give the individual user access (od-user).

To assign or revoke system roles, follow these steps:

1. Select **User Access > Roles** to display the System Roles window.
2. (Windows only) Select the domain containing the individual's system account from the **Domain** list.
3. Enter the individual's user name in the **Username** box and click **Lookup User**. The user name you enter must be valid user account on the operation server host.

   If the individual is new to the OpenDeploy environment (does not yet have any roles assigned on the operations server), then both the `od-admin` and `od-roles` options will be listed in the **Available Roles** list. If the individual already has a system role assigned, that role will be displayed in the **Authorized Roles** list.
4. Select the role you want to assign the individual, and click **Add** to move that role to the **Authorized Roles** list.
5. Select any role currently held by the individual from the **Authorized Roles** list, and click **Remove** to move that role to the **Available Roles** list. That individual will no longer have that role access to the OpenDeploy environment.
6. Repeat this procedure for every individual to whom you want to assign or revoke a system role.

After you have established all your system roles, you can assign server roles to each individual.

Specifying Server Role Access

Specifying the server role access of an individual determines what level of access that individual will have on a particular OpenDeploy host server. The type of role available to assign an individual is determined by what level of system role the individual was granted. An individual assigned with a system role of User (`od-user`) can only be assigned a User role (`od-user`) on a particular host. An individual assigned a system role of Administration (`od-admin`) can be assigned an Administrator role (`od-admin`) on a particular host. An individual with an Administrator or User system role, but no server role, assigned for a particular server can monitor deployments on that OpenDeploy server. However, the individual cannot perform tasks that require an Administrator or User role's authentication.

You can assign and manage server role access through the user interface in the Server Access window (Figure 18).

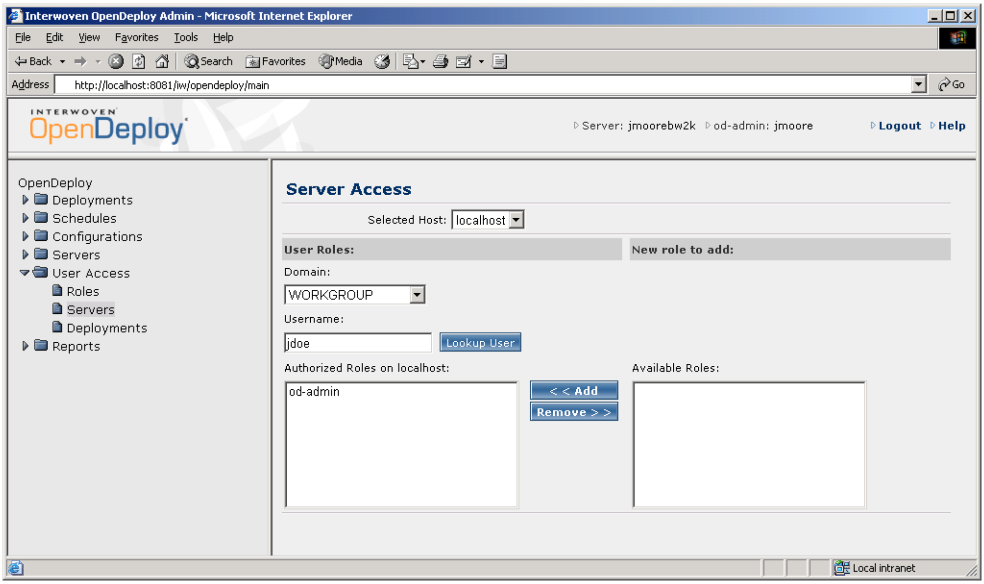


Figure 18: Server Access Window

Assigning or revoking server role access is similar to assigning system role access, except it is done on a per-server basis. You must assign and revoke roles for each server separately.

To assign or revoke server roles, follow these steps:

1. Select **User Access > Servers** to display the Server Access window.
2. Select the OpenDeploy host to which you are assigning roles from the **Selected Host** list.
3. (Windows only) Select the domain containing the individual's system account from the **Domain** list.

4. Enter the individual's user name in the **Username** box and click **Lookup User**.

   If the individual is new to that particular OpenDeploy host (does not yet have any roles assigned), then the available role options are listed in the **Available Roles** list, depending on what system roles the individual is assigned:

   – Administrator — `od-admin` is listed.

   – User — `od-user` is listed.

   If the individual already has a role assigned on that OpenDeploy host, that role will be displayed in the **Authorized Roles** list.

5. Select any role you want to assign the individual from the **Available Roles** list, and click **Add** to move that role to the **Authorized Roles** list.

6. Select any role currently held by the individual from the **Authorized Roles** list, and click **Remove** to move that role to the **Available Roles** list. That individual will no longer have that role access to the host server.

7. Repeat this procedure for every individual to whom you want to assign or revoke a server role.

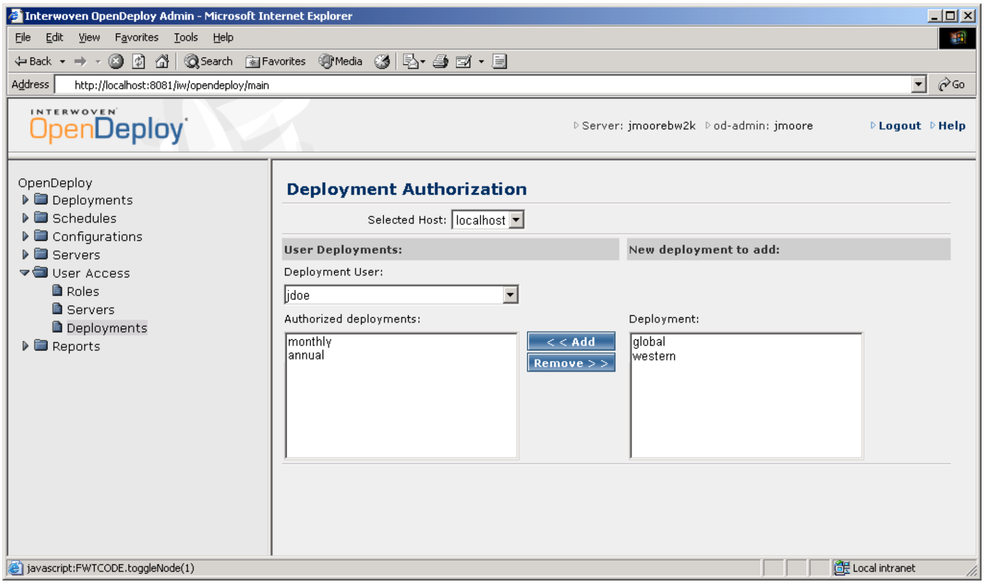
Assigning User Roles Deployments

Individuals with administrator access to an OpenDeploy host have unlimited access to that host's deployments. An administrator can limit the access of an individual with a User role on that host to only those deployments the administrator assigns. An individual with User role access can perform the following tasks on that host:

- Start and cancel deployments for which the individual is authorized.
- View the XML-based configuration for which the individual is authorized.
- Create and edit schedules for which the individual is authorized.
- Monitor the status of all deployments.
- View the schedules for all deployments.

The same deployment can be assigned to more than one individual with User role access on that host, and those individuals can have any number of deployments assigned them.

You can authorize role access to specific deployments on an OpenDeploy host through the user interface in the Deployment Authorization window (Figure 19).


Interwoven OpenDeploy Admin - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8081/iw/opendeploy/main
Go
INTERWOVEN OpenDeploy
Server: jmoorebw2k
od-admin: jmoore
Logout
Help
OpenDeploy
Deployments
Schedules
Configurations
Servers
User Access
Roles
Servers
Deployments
Reports
<< Add
Remove >>
javascript:FWTCODE.toggleNode(1)
Local intranet

Deployment Authorization

Selected Host: localhost

User Deployments:

Deployment User: jdoe

Authorized deployments: monthly, annual

New deployment to add:

Deployment: global, western

Figure 19: Deployment Authorization Window

To authorize individuals with User roles to perform specified deployments on a particular OpenDeploy host, follow these steps:

1. Select **User Access > Deployments** to display the Deployment Authorization window.
2. Select the OpenDeploy host server whose User roles you want to modify from the **Selected Host** list.
3. Select the user for whom you are authorizing deployments from the **Deployment User** list.

   This list will display those individuals who have User or Administrator role access to the OpenDeploy host you selected. Those deployments for which the individual already has authorization to use are displayed in the **Authorized Deployments** list. Those deployments available to be assigned are displayed in the **Deployment** list.

4. Select any deployment you want to disallow the user to control from the **Authorized Deployments** list and click **Remove**. Those deployments are now displayed in the **Deployment** list.
5. Select any deployment you want to authorize the user to perform from the **Deployment** list and click **Add Deployment**. That deployment now appears in the **Authorized Deployments** list.

Role Access in TeamSite Workflows

When including an external script to run OpenDeploy in a TeamSite workflow, the user that is running the OpenDeploy task must have the correct permissions to run the deployment in the task. If the user does not have the correct deployment permissions the deployment can fail.

Composing Deployments

You can create a new deployment configuration file, or edit an existing one, using the following methods:

- Using a text or XML Editor
- Using the Deployment Configuration Composer

Using a Text or XML Editor

Because deployment configuration files are XML-based, you can use your favorite text or XML editor to open and modify any of them. Using a text or XML editor to edit configuration files requires you to have file system access to the OpenDeploy host where the deployment you want to create or modify resides. All new and modified deployment configuration files must reside in the following location:

```
od-home/conf
```

for OpenDeploy to use them.

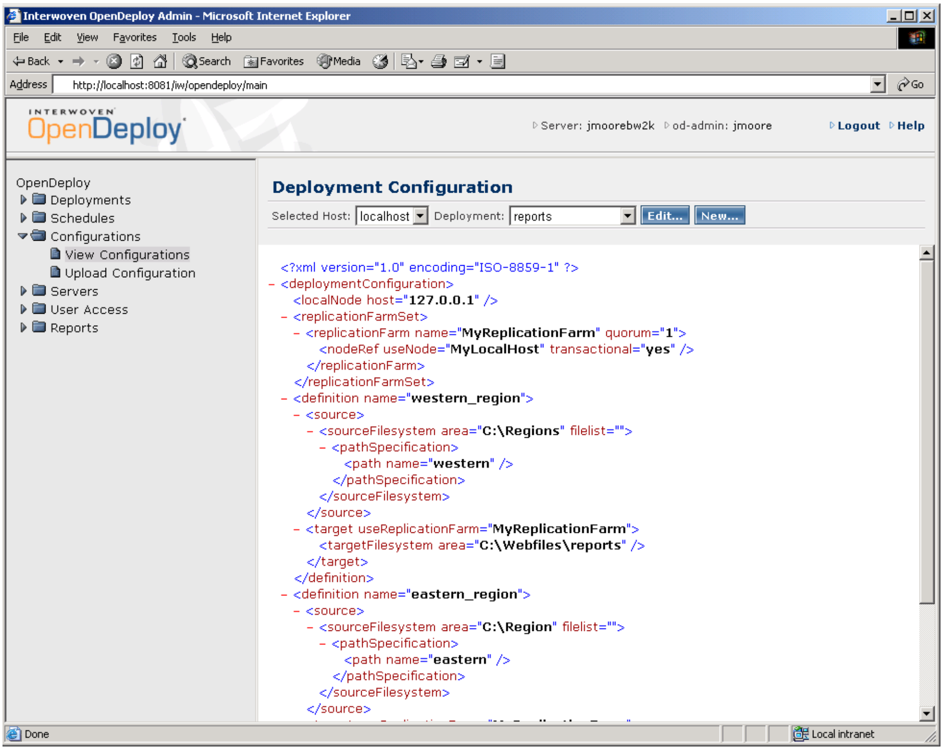
Modifying deployment configuration files requires an understanding of XML syntax, and of the deployment configuration DTD. You should not try to modify deployment configuration file unless you have expertise in both. See Chapter 3, “Deployment Types” for more information on modifying deployment configuration files.

Using the Deployment Configuration Composer

The Deployment Configuration Composer is a tool in the OpenDeploy user interface that allows you to create and modify existing deployment configurations without having to edit the files using a text or XML editor. Knowledge of XML is not required to use this tool. However, you do need to understand the OpenDeploy features and functionality described in Chapter 3, "Deployment Types" and Chapter 4, "Deployment Features" before you can create a complete deployment configuration. See Chapter 9, "Composing Deployments" for more information on how to use this tool.

Viewing Deployment Configuration Source Code

You can view the XML-based source code of a selected deployment configuration's file (Figure 20) within the OpenDeploy user interface.



Deployment Configuration

Selected Host: localhost Deployment: reports

```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <deploymentConfiguration>
    <localNode host="127.0.0.1" />
  - <replicationFarmSet>
    - <replicationFarm name="MyReplicationFarm" quorum="1">
        <nodeRef useNode="MyLocalHost" transactional="yes" />
      </replicationFarm>
    </replicationFarmSet>
  - <definition name="western_region">
    - <source>
      - <sourceFilesystem area="C:\Regions" filelist="">
        - <pathSpecification>
            <path name="western" />
          </pathSpecification>
        </sourceFilesystem>
      </source>
    - <target useReplicationFarm="MyReplicationFarm">
        <targetFilesystem area="C:\Webfiles\reports" />
      </target>
    </definition>
  - <definition name="eastern_region">
    - <source>
      - <sourceFilesystem area="C:\Region" filelist="">
        - <pathSpecification>
            <path name="eastern" />
          </pathSpecification>
        </sourceFilesystem>
      </source>
```

Figure 20: View Configuration Window

Viewing the source code of a deployment configuration allows you to identify and troubleshoot a deployment configuration file issue quickly. You can also use this feature simply to see what element and attribute values are present. However, you cannot actually edit a deployment configuration displayed in the Deployment Configuration window. Instead you must edit the deployment configuration either using a text or XML editor, or using the Deployment Configuration Composer. See "Composing Deployments" on page 83 for more information.

To view the source code of a deployment configuration, follow these steps:

1. Select **Configurations > View Configuration** to display the View Configuration window.
2. Select the name of the OpenDeploy server hosting the deployments whose configuration information you want to view from the **Selected Host** list.
3. Select the name of the deployment whose configuration information you want to view from the **Deployment** list. This information is displayed in the window below.

You can also view the configuration of a selected deployment in the Start Deployment window by clicking **View Config**.

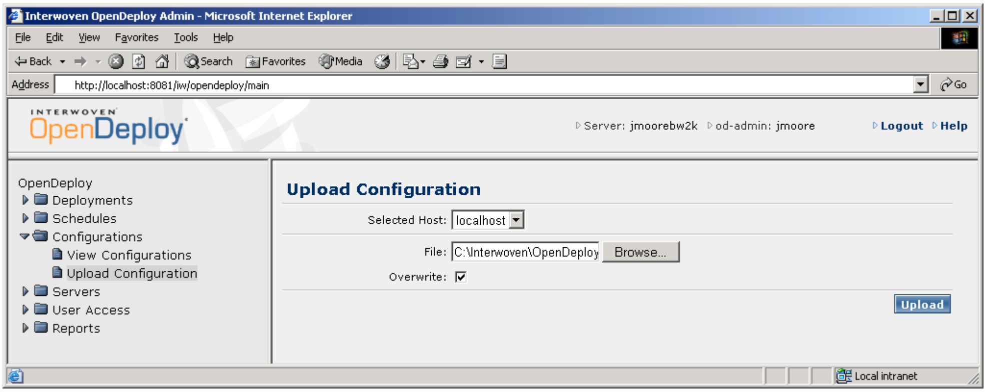
Uploading Deployment Configurations

The required location for deployment configuration files is:

```
od-home/conf
```

Deployment configuration files you place in this location will appear in the Start Deployment window's **Deployment** list as a selectable deployment. You can run this deployment from the user interface or the command line, assuming that the deployment conforms to the XML syntax and deployment configuration DTD rules, and that individuals in the User role have the proper access.

You can upload deployment configurations through the Upload Configuration window (Figure 21).



Selected Host: localhost

File: C:\Interwoven\OpenDeploy

Overwrite: ☑

Figure 21: Upload Configuration Window

This feature will copy a configuration from any file system location into the *od-home*/conf directory. You must have Administrator role access to import deployment configurations in this manner. Individuals with User role access will be denied this feature. See “Roles and Authorization” on page 75 for more information.

To upload a deployment configuration, follow these steps:

1. Select **Configurations > Upload Configuration** to display the Upload Configuration window.
2. Select the name of the OpenDeploy server receiving the imported deployment configuration from the **Selected Host** list.
3. Enter the path to the file you want to import. You can also click **Browse** and navigate to the location of the file.
4. Check the **Overwrite** box if you are overwriting an existing deployment configuration file.
5. Click **Upload**. The file you uploaded is now available for use.

Running Deployments from the User Interface

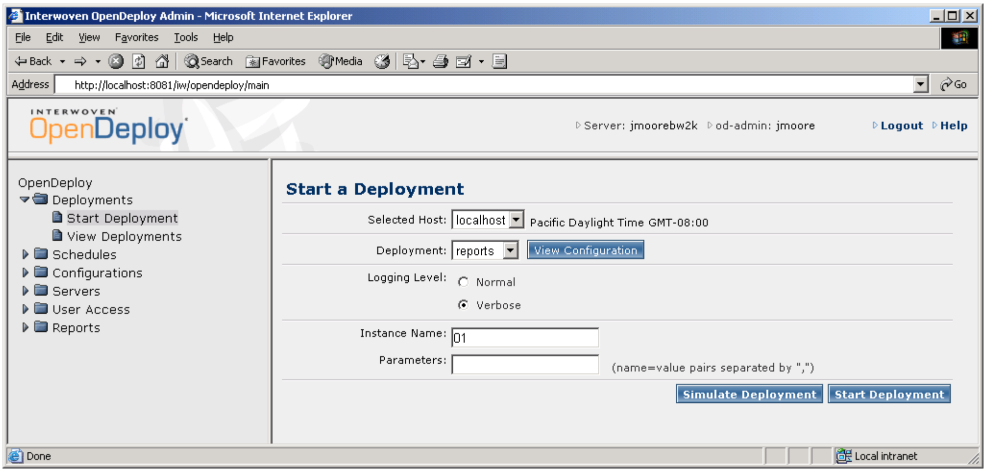
This section describes how start and cancel deployments using the browser-based user interface. For instructions on running deployments from the command-line, see "Running Deployments from the Command Line" on page 101.

An individual holding an Administrator role on the OpenDeploy server can start and cancel any deployment on that host. Individuals holding User roles on that host only can start and cancel deployments on that host that are assigned to them. Individuals holding either type of role can view the progress and status of any deployment.

Starting a Deployment

You should perform a test deployment using the `test.xml` deployment configuration before trying any other deployments. Performing the test will ensure that your OpenDeploy server is properly configured, and will give you practice with deployments. See "Performing a Test Deployment" on page 90 for more information.

You can start a deployment through the OpenDeploy user interface (Figure 22).



Start a Deployment

Selected Host: localhost Pacific Daylight Time GMT-08:00

Deployment: reports

Logging Level: ○ Normal ● Verbose

Instance Name: 01

Parameters: (name=value pairs separated by ",")

Figure 22: Start a Deployment Window

You can also start a deployment by entering the appropriate command-line tool at the command prompt.

To start a deployment using the OpenDeploy user interface, follow these steps:

1. Select **Deployments > Start Deployment** to display the Start Deployment window.

2. Select the OpenDeploy server you want to act as the source of the deployment from the **Selected Host** list. When you select a particular host, its deployment choices become available in the **Deployment** list.

3. Select the deployment you want to start from the **Deployment** list. If you are performing an initial test of OpenDeploy, select **`test`**.

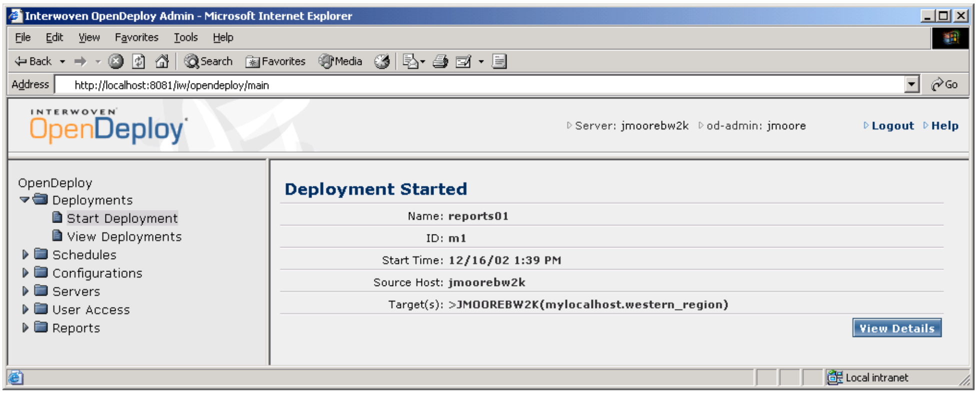
4. Select your choice from one of the following **Logging Level** options:
   - **Verbose** — logs high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
   - **Normal** — logs standard status and error messages. In most cases, this level of logging provides a sufficient amount of detail to meet your needs.

5. (Optional) Enter the instance name of the deployment in the **Instance Name** box. The use of instances is described later in this section.

6. (Optional) Enter the name=value pair for the parameter substitution in the **Name=Value** box. If you are adding multiple name=value pairs, separate each one with a comma (","). See "Parameter Substitution" on page 198 for more information on this feature.

7. Click **Simulate Deployment** if you want to perform a simulated deployment before actually moving files to the target hosts. Otherwise, go on the next step. See "Performing a Simulated Deployment" on page 91 for more information.

8. Click **Start Deployment** to start the deployment. The Deployment Started window appears (Figure 23), displaying information regarding the deployment you just started.



Deployment Started

Name: reports01

ID: m1

Start Time: 12/16/02 1:39 PM

Source Host: jmoorebw2k

Target(s): >JMOOREBW2K(mylocalhost.western_region)

Figure 23: Deployment Started Window

Deployment Instance Naming

You can append the deployment name with a name, number, or some other identifying characters to create a unique instance of that deployment. This allows a deployment using the same configuration file to be run using a different deployment name. When you specify multiple instances of a deployments in this manner, they can run simultaneously. If instance name is not used, the deployment can only be run once, and a new running of the deployment cannot be started until the previous one has finished.

A common use of this feature is in situations where there are multiple users initiating the same deployment in an uncoordinated fashion (such as each running a workflow). By specifying a different instance for each running of the deployment, you can ensure all the deployment jobs get run.

You can specify an instance in the **Instance Name** box of the Start a Deployment window. The value you specify for the instance can be any combination of identifying characters. Spaces are not permitted in the instance name. Typically, instance names are numbers or a short descriptive term such as `01` or `Monthly`.

The deployment name and the instance name you specify are combined together to form the complete instance name. For example:

`reports01` *or*

`reportsMonthly`

No delimiter character is included, although you can specify one as part of the instance name itself, such as a period (`.`) or underscore (_). This delimiter character appears in the complete instance name:

`reports_01` *or*

`reports.monthly`

When you run or schedule a deployment with an instance, OpenDeploy appends the instance to the deployment name and uses that extended name in the browser bases user interface, logging, and anywhere else where the deployment is tracked or monitored.

Performing a Test Deployment

After you have installed and configured OpenDeploy, you should perform a test deployment to ensure everything is working correctly. The OpenDeploy software includes a test deployment configuration `test.xml` that will deploy files from one location to another on your OpenDeploy host. The test configuration only uses default settings present in the host configurations files, so no further configuration is required on your part.

The test deployment configuration will move the following file:

`od-home``/conf/dtd/oddeployment.dtd`

to the following location:

`od-home``/tmp`

Use either the OpenDeploy user interface or command-line method to start the deployment. See "Starting a Deployment" on page 87 for more information. If you successfully deployed the file to its designated target location on your server, then you are ready to perform a deployment to a target on another host.

Performing a Simulated Deployment

You can perform a simulated deployment of any deployment configuration using the *Simulate Deployment* feature. The Simulate Deployment feature performs a simulated deployment that does not actually transfer any content over to the target, but logs what would have been transferred over. Using this feature allows you to test out and see what would have been transferred if the deployment was actual. The record of this result is in the deployment log files.

To perform a simulated deployment, follow these steps:

1. Select **Deployments > Start Deployment** to display the Start Deployment window.
2. Select the OpenDeploy server you want to host the simulated deployment from the **Selected Host** list. When you select a particular host, its deployment choices become available in the **Deployment** list.
3. Select the deployment you want to simulate from the **Deployment** list.
4. Select your choice from one of the **Logging Level** options. This is the level of logging that will be performed for your simulated deployment. See "Starting a Deployment" on page 87 for more information.
5. Click **Simulate Deployment** to begin the simulation. The Deployment Started window appears.
6. Click **View Details** to display the View Deployments window. Here you can view deployment information for your simulated deployment just like an actual deployment.
7. Click **View Log** to view logging information. Here you can view the list of files that would have been included in an actual deployment.

After evaluating the simulated deployment, you can actually deploy the files to the target hosts by clicking **Start Deployment**.

Checking File Integrity on Production Servers

The Simulate Deployment feature can also serve as a valuable tool in ensuring the integrity of Web site files residing on your production servers. You can use this feature to determine if a targeted production server is out of sync with your development server. Running the Simulate Deployment feature will create an entry for any file that would be deployed in the deployment log file. A file that was deployed unexpectedly is indicative of a file being mistakenly or intentionally added, deleted, or changed. Use the Simulate Deployment feature regularly to ensure the integrity of your production server Web sites. See "Performing a Simulated Deployment" on page 91 for information on using the Simulate Deployment feature.

Cancelling Deployments in Progress

You can cancel a deployment in progress during certain stages of the deployment process depending on the type of deployment:

- Initial setup — all deployments
- Compare phase — deployments that compare files only
- Pre-commit phase — transactional deployments only

If you choose to cancel a deployment during one of these phases, OpenDeploy will complete the phase in progress and then cancel the remainder of the deployment. In some cases, a lengthy amount of time can pass between the time you order the cancellation and when OpenDeploy completes the phase in progress before quitting the deployment. You cannot cancel a deployment once the pre-commit phase is over.

Initial Setup

All deployments take time to set up the deployment before files are actually compared or moved. A larger deployment with more target hosts take longer to perform its initial setup than a smaller deployment with a lower number of targets. Any deployment can be canceled during its setup phase.

Compare Phase

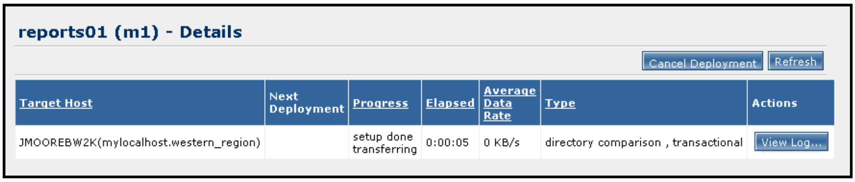
The *compare phase* is when OpenDeploy compares the files between file system locations or TeamSite areas. Those deployments that compare files can be canceled during their compare phases as well as their initial setup. The length of the compare phase is dependent on the number of files being compared. A small number of files in a deployment, even if their total file size is large, will result in a brief compare phase. A large number of files, even if the total file size is smaller, will result in a longer compare phase.

Deployment types that do not compare files, such as file list deployments, do not have a compare phase. These can only be canceled during their initial setup.

Pre-Commit Phase

Transactional deployments can be canceled before or during their *pre-commit phase* in addition to their initial setup and compare phases. The pre-commit phase is when OpenDeploy determines whether or not all the targets have successfully received their deployments.

The **Details** table in the Source Deployments window contains the **Cancel Deployment** button (Figure 24).



reports01 (m1) - Details

Target Host	Next Deployment	Progress	Elapsed	Average Data Rate	Type	Actions
JMOOREBW2K(mylocalhost.western_region)		setup done transferring	0:00:05	0 KB/s	directory comparison , transactional	View Log...

Figure 24: Details Table with Cancel Deployment Button

This button is active when cancellation of a running deployment is permitted. If the deployment has progressed past that time, or if it is completed, the button is disabled. Clicking **Cancel Deployment** stops the deployment at that point. In some cases, the deployment cancellation window is too short to permit cancellation of the deployment. See "Monitoring Deployments" on page 95 for more information on the **Details** table.

You can only cancel a deployment in progress from the source host using the OpenDeploy user interface. A target host cannot cancel a deployment it is receiving. There is no method for cancelling a deployment from the command prompt.

You cannot restart a deployment after it has been cancelled. If a transactional deployment has been cancelled, the deployment is considered to have failed and is rolled back with the targets restored to their previous states.

Depending on the speed of the deployment phases and when you issue the cancellation order, it is possible that a deployment you attempt to cancel will be completed anyway. The time when a cancellation is possible might close before OpenDeploy can process the deployment cancellation order after you click **Cancel Deployment**. In other cases, the cancellation window can close before the user interface can fully display the Details table with the **Cancel Deployment** button displayed.

To cancel a deployment in progress sent by your host, follow these steps:

1. Select **Deployments > View Deployments** after a deployment has started to display the Source Deployments window.

   If you started the deployment manually, you can click **View Details** in the Deployment Started window to display the Source Deployments window and the **Details** table for your deployment. Skip to step 4.

2. Select the OpenDeploy server whose deployment you want to cancel from the **Selected Host** list.

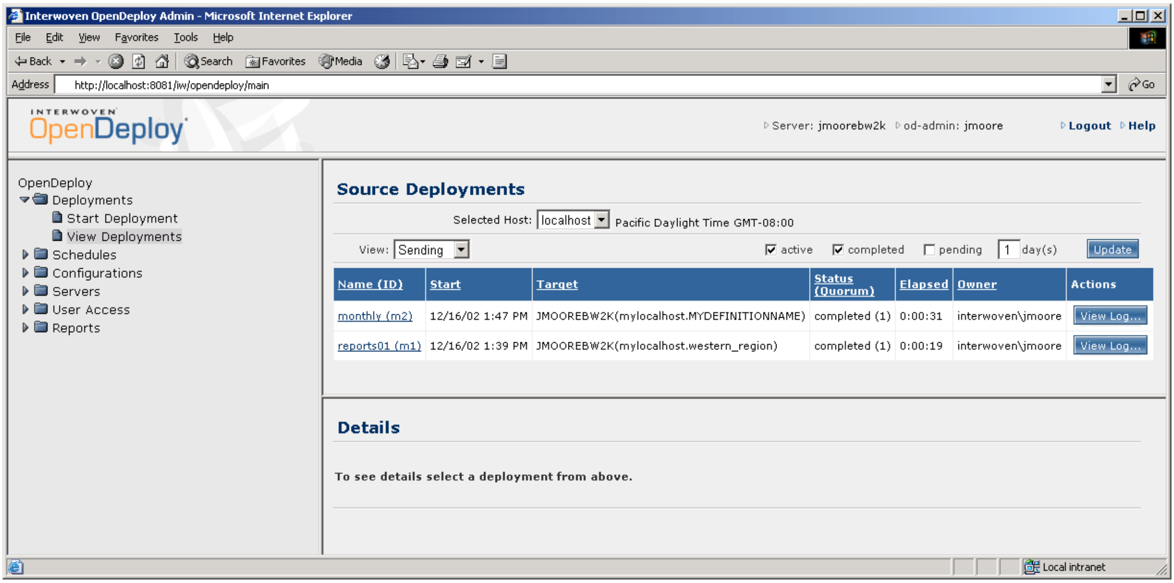
3. Select the link of the deployment you want to cancel. That deployment's target hosts are displayed in the **Details** table.

   If the cancellation window for the deployment is still open, the **Cancel Deployment** button is displayed for that target.

4. Click **Cancel Deployment** to stop the deployment for each target host you want.

Monitoring Deployments

You can view information regarding the deployments being sent in the Source Deployments window (Figure 25) and received in the Target Deployments window. These windows include information on deployments that have already taken place, that are currently in progress, and that are pending. You can also access log information and other details on a specific deployment.



Source Deployments

Selected Host: localhost Pacific Daylight Time GMT-08:00

View: Sending ☑ active ☑ completed ☐ pending 1 day(s) Update

Name (ID)	Start	Target	Status (Quorum)	Elapsed	Owner	Actions
monthly (m2)	12/16/02 1:47 PM	JMOOREBW2K(mylocalhost.MYDEFINITIONNAME)	completed (1)	0:00:31	interwoven\jmoore	View Log...
reports01 (m1)	12/16/02 1:39 PM	JMOOREBW2K(mylocalhost.western_region)	completed (1)	0:00:19	interwoven\jmoore	View Log...

Details

To see details select a deployment from above.

Figure 25: Source Deployments Window

To monitor the progress of your deployments, follow these steps:

1. Select **Deployments > View Deployments** to display the Source Deployments window.
2. Select the OpenDeploy server whose deployments you want to monitor from the **Selected Host** list.

3. Select one of the following choices from the **View** list:
   - **Sending** — select to display the Source Deployments window. Here information on deployments initiated by the host server is displayed. See the following section for details on the contents of the Source Deployments window.
   - **Receiving** — select to display the Target Deployments windows. Here information on deployments received by the host server is displayed. See the following section for details on the contents of the Target Deployments window.

Viewing Options

The viewing options are similar for both the Source Deployments and Target Deployments windows:

- **Active** check box — check to view deployments that are currently active.
- **Completed** check box — check to view deployments that have been completed. The number of deployments displayed can be modified. See "Completed Sent Deployments Limit" on page 99 and "Completed Received Deployments Limit" on page 100 for more information.
- **Pending** check box (Source Deployments window only) — check to view scheduled deployments that have not occurred yet. You can specify the deployments pending to a specified number of days in the future by entering that number in the text box.
- **Update** button — click to refresh the window after changing the viewing options.

Deployments Table

The **Deployments** table displays the following information regarding all deployments being sent or received by the selected host, depending on whether the Source and Target Deployments window is being displayed.

- **Name (ID)** column — displays the name and identification number of the deployment. If an instance value was specified at the time the deployment was run or scheduled, the deployment name is appended with that instance. For example, if you ran the deployment `reports` and specified the instance value `01`, that deployment would be displayed as `reports01`.

On the receiver, the name value is:

deployment.definition.target-server

where the following variables apply:

– *deployment* — the name of the associated deployment.

– *definition* — the name of the definition in the deployment configuration that contains the source/target pairing.

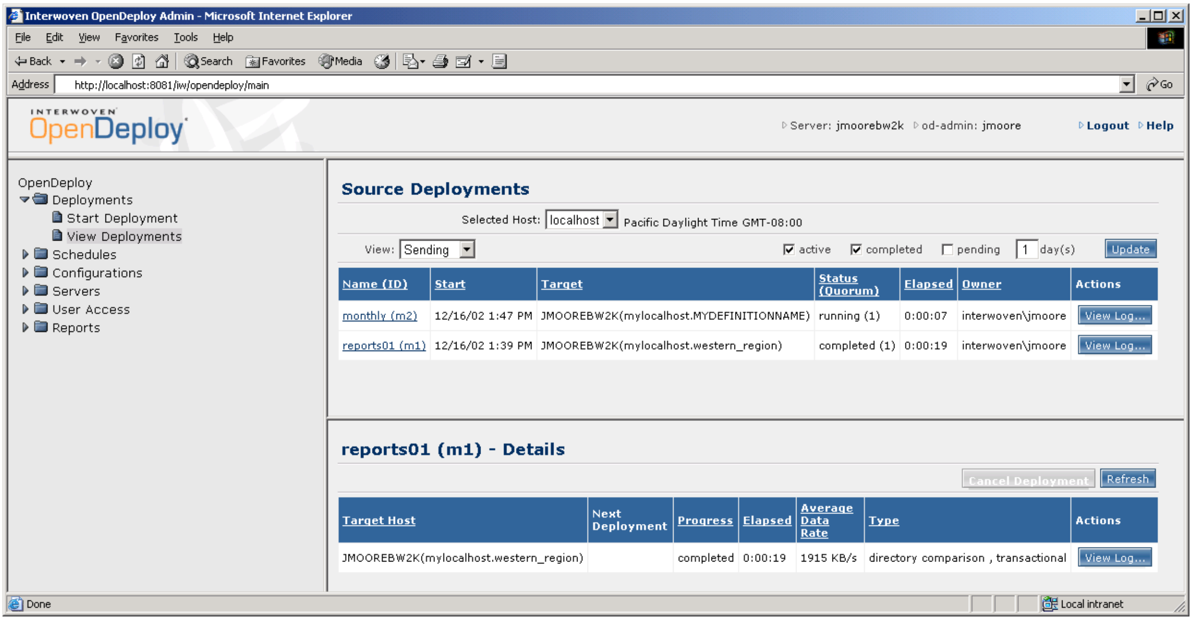
– *target-server* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.

Selecting a deployment name displays its details in the **Details** table.

- **Start** column — displays the date and time the deployment started.
- **Target** column (Source Deployments window only) — displays the name of each target host receiving the deployment.
- **Source** column (Target Deployments screen only) — displays the name of the source host sending the deployment.
- **Status** column — displays the current status of the deployment, such as whether it is running, pending, completed, or failed.
- **Elapsed** column — displays the elapsed time the deployment has been running.
- **Owner** column — displays the login name of the deployment's owner.

Details Table

Clicking the deployment name displays the **Details** table underneath the **Deployments** table (Figure 26).



Source Deployments

Selected Host: localhost Pacific Daylight Time GMT-08:00

View: Sending ☑ active ☑ completed □ pending 1 day(s)

Name (ID)	Start	Target	Status (Quorum)	Elapsed	Owner	Actions
monthly (m2)	12/16/02 1:47 PM	JMOOREBW2K(mylocalhost.MYDEFINITIONNAME)	running (1)	0:00:07	interwoven\jmoore	View Log...
reports01 (m1)	12/16/02 1:39 PM	JMOOREBW2K(mylocalhost.western_region)	completed (1)	0:00:19	interwoven\jmoore	View Log...

reports01 (m1) - Details

Target Host	Next Deployment	Progress	Elapsed	Average Data Rate	Type	Actions
JMOOREBW2K(mylocalhost.western_region)		completed	0:00:19	1915 KB/s	directory comparison , transactional	View Log...

Figure 26: Source Deployments Window — Details Table

The **Details** table displays information regarding the source host or target hosts participating in the deployment you selected. If the Source Deployments window is displayed and you select a deployment with multiple target hosts, each of those target hosts will be displayed in the **Details** table.

- **Target Host** (Source Deployments window only) — displays the name of the server receiving the deployment as it appears in the nodes configuration file of the sending host.
- **Source Host** (Target Deployments window only) — displays the name of the server sending the deployment.
- **Progress** column — displays the status and particular activity taking place regarding the deployment at that given time.
- **Elapsed** column — displays the elapsed time the deployment has been running.

- **Average Data Rate** column — displays the average data transmission rate of the deployment at that given time.
- **Type** column— displays the type of deployment, such as directory comparison, TeamSite comparison, file list, and others.

Click **Refresh** to update the **Details** table with the latest information on the selected deployment, such as whether a deployment in progress has completed yet.

If you display the Source Deployments window while the deployment is in progress, the **Details** table will indicate the deployment is still in progress, and under certain circumstances gives you the option of cancelling the deployment. See "Cancelling Deployments in Progress" on page 92 for more information.

Source Deployments Window

The Source Deployments window (Figure 25) appears when you select **Sending** from the **View** list. The Source Deployments window contains information regarding deployments originating from the selected OpenDeploy host that are displayed in the **Deployments** table.

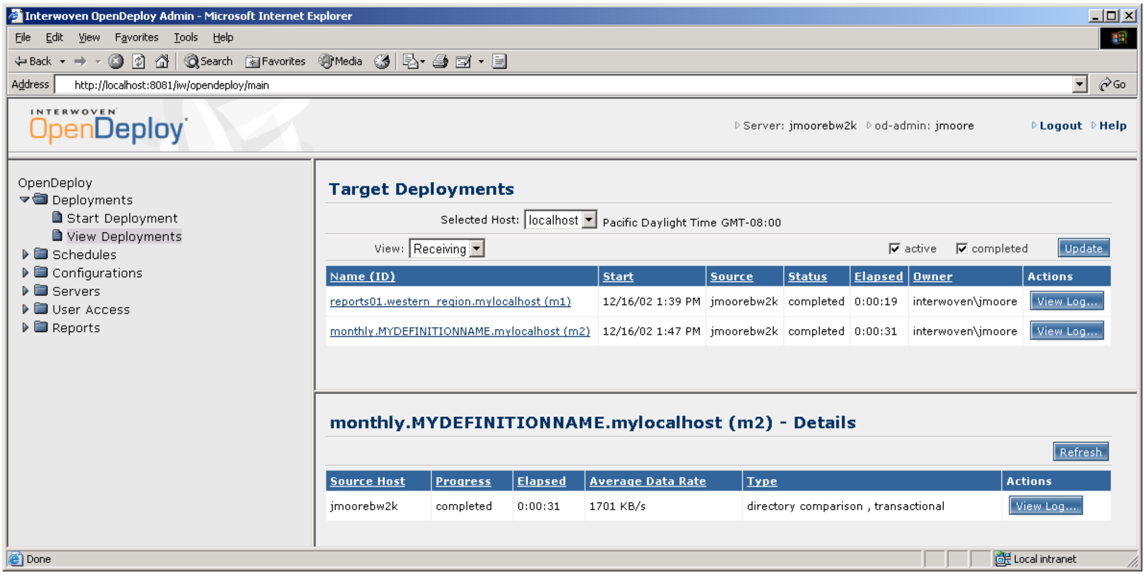
The deployment name and its ID are displayed in the **Name (ID)** column of the **Deployments** table, along with other information about the deployment. If an instance value was specified at the time the deployment was run or scheduled, the deployment name is appended with the instance value. See "Deployments Table" on page 96 for a description of the **Deployments** table contents.

Completed Sent Deployments Limit

By default, the Source Deployments window displays the last 50 deployment jobs completed (when the **Completed** option is selected) that were sent by the selected host. This number includes multiple instances of the same deployment configuration being run. You can adjust that number in the host's base server configuration file. Refer to "Specifying the Completed Deployments List" on page 82 in the *OpenDeploy Installation and Reference Guide* for more information.

Target Deployments Window

The Target Deployments window (Figure 27) appears when you select **Receiving** from the **View** list.



Target Deployments

Selected Host: localhost Pacific Daylight Time GMT-08:00

View: Receiving ☑ active ☑ completed

Name (ID)	Start	Source	Status	Elapsed	Owner	Actions
reports01.western_region.mylocalhost (m1)	12/16/02 1:39 PM	jmoorebw2k	completed	0:00:19	interwoven\jmoore	View Log...
monthly.MYDEFINITIONNAME.mylocalhost (m2)	12/16/02 1:47 PM	jmoorebw2k	completed	0:00:31	interwoven\jmoore	View Log...

monthly.MYDEFINITIONNAME.mylocalhost (m2) - Details

Source Host	Progress	Elapsed	Average Data Rate	Type	Actions
jmoorebw2k	completed	0:00:31	1701 KB/s	directory comparison , transactional	View Log...

Figure 27: Target Deployments Window

The Target Deployment window contains information regarding deployments being received by the selected OpenDeploy host. Like the Source Deployments window, selecting a deployment from the **Name (ID)** column displays additional information in the **Details** table underneath.

Completed Received Deployments Limit

By default, the Target Deployments window displays the last 50 deployment jobs completed (when the **Completed** option is selected) that were received by the selected host. This number includes multiple instances of the same deployment configuration being run. You can adjust that number in the host's base server or receiver configuration file. Refer to "Specifying the Completed Deployments List" on page 82 in the *OpenDeploy Installation and Reference Guide* for more information.

Deployment Logging

Clicking **View Log** for either a deployment listed in the Source Deployment window, or one of the deployment's corresponding source or target host listings in the **Details** table will display various types of logging information. See Chapter 6, "Logging" on page 243 for more information.

Running Deployments from the Command Line

Command-line tools only can be issued on the host where the OpenDeploy server is installed. Commands can be issued by anyone regardless of whether they hold an Administrator or User role. There are no authentication or authorization checks on individuals issuing these commands.

You can start a deployment and perform associated tasks using the `iwodstart` command. There are various options associated with the `iwodstart` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodstart -h | -v

iwodstart deployment [-async] [-inst instance] [-k "key=value"]+ [-sim]
[-V (normal | verbose)]
```

`-h` Displays usage information.

`-v` Displays version information.

`deployment` Name of the deployment to start.

`-async` Runs `iwodstart` command asynchronously. The `iwodstart` command will return before the deployment completes. See "Running Deployments Asynchronously" on page 104 for more information.

`-inst` Includes the deployment instance name *`instance`*, which is a suffix that is appended to the deployment name. This option is used to create unique deployment names for each instance of a deployment configuration. See "Specifying a Deployment Instance" on page 103 for more information.

`-k` *`arg`*	Key/value substitution with `"`*`key=value`*`"` as the *`arg`* value. See “Parameter Substitution” on page 198 for more information.
`-sim`	Enables the simulated deployment feature. See “Performing a Simulated Deployment” on page 91 for more information.
`-V` *`arg`*	Logging level with `verbose` or `normal` as args. See “Defining Logging Levels from the Command Line” on page 258 for more information.

The `iwodstart` command returns the following codes regarding the status of the deployment:

- `0` — succeeded
- `1` — failed to start
- `2` — ran and returned a failed status
- `9` — completed with errors

Starting a Deployment

To start a deployment from the command line, follow these steps:

1. Navigate to the following directory:

   `od-home``/bin`

2. Start the deployment by entering the following command at the prompt:

   ```
   iwodstart deployment
   ```

   where *`deployment`* is the name of the deployment you want to start. For example, if you wanted to run the deployment *reports*, you would enter the following command at the prompt:

   ```
   iwodstart reports
   ```

Performing a Simulated Deployment

See “Performing a Simulated Deployment” on page 91 for a description of the simulated deployment feature and its application.

Performing a simulated deployment from the command-line requires adding the `-sim` option to the `iwodstart` command. For example, if you wanted to run the deployment *reports* as a simulated deployment, you would enter the following command at the prompt:

```
iwodstart reports -sim
```

Specifying a Deployment Instance

Deployments are appended using the `iwodstart` command with the `-inst` *`instance`* option using the following syntax:

```
iwodstart deployment -inst instance
```

The value you specify for *`instance`* can be any combination of identifying characters. Spaces are not permitted in the instance name. Typically, instance names are numbers or a short descriptive term. For example:

`iwodstart reports -inst 01` *or*

`iwodstart reports -inst Monthly`

The deployment name and the instance name you specify are combined together to form the complete instance name. For example:

`reports01` *or*

`reportsMonthly`

No delimiter character is included, although you can specify one as part of the instance name itself, such as a period (`.`) or underscore (_). For example:

`iwodstart reports -inst _01` *or*

`iwodstart reports -inst .monthly`

This delimiter character appears in the complete instance name:

```
reports_01 or
```

```
reports.monthly
```

See "Deployment Instance Naming" on page 89 for more information on this feature.

Use with Parameter Substitution

The deployment instance feature is often used with the parameter substitution, which allows you to run a single deployment while specifying different parameter values for each instance. See "Parameter Substitution" on page 198 for more information on this feature, including examples on using the instance feature.

Use with Schedules

You can schedule deployments using the instance feature. See "Scheduling Deployment Instances" on page 235 for more information on this usage.

Running Deployments Asynchronously

In some situations, you may want to start a deployment but not wait for the deployment to end before moving on to other tasks.This is known as an asynchronous deployment. For example, you may have a script that starts the deployment and then proceeds to other tasks without waiting to determine whether the deployment completed.

When a deployment is run asynchronously, only the deployment's success or failure to start is returned. No indication of the deployment's success or failure to complete is presented. Instead, you must use another method to determine the status of the deployment, such as reviewing the log files or deployment reports.

You can run a deployment asynchronously using the `-async` option. For example, if you wanted to run the deployment *reports* asynchronously, the command would be:

```
iwodstart reports -async
```

When you include the `-async` option, the `iwodstart` command returns as soon as the deployment starts. Without the `-async` option, `iwodstart` would wait for completion of the deployment before returning.

Cancelling a Deployment in Progress

You can cancel a deployment in progress from the command line using the `iwoddeploycancel` command-line tool.

There are various options associated with the `iwoddeploycancel` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwoddeploycancel -h | -v

iwoddeploycancel deployment
```

`-h`	Displays usage information.
`-v`	Displays version information.
`deployment`	Name of the deployment to cancel.

To cancel a deployment in progress from the command line, follow these steps:

1. Navigate to the following directory:

   `od-home``/bin`

2. Cancel the deployment by entering the following command at the prompt:

   `iwoddeploycancel` ***`deployment`***

   where *deployment* is the name of the deployment running you want to cancel in progress. For example, if you wanted to cancel the deployment *reports*, you would enter the following command at the prompt:

   `iwoddeploycancel reports`

   The deployment is stopped with no further activity.

The `iwoddeploycancel` command-line tool follows the same criteria for cancelling a deployment in progress as the browser-based user interface. See “Cancelling Deployments in Progress” on page 92 for more information.

Chapter 3

Deployment Types

This chapter describes the different types of deployments, and how to configure deployment configuration files.

Deployment Configuration Files

A deployment configuration file is an XML-based file containing elements and attributes that define how the deployment will work. Some elements and attribute values are required, while others are optional. In most cases, if an optional value is not specified in the deployment configuration file, OpenDeploy will use a built-in default value. In some cases, OpenDeploy will look to the host's base server or receiver configuration files for setting information if none exists in the deployment configuration file. The rest of this chapter discusses features available in OpenDeploy by modifying the deployment configuration file. You should have some knowledge of XML before modifying these files.

Understanding the Configuration DTDs

OpenDeploy configuration files are XML files based on the rules defined in the corresponding configuration DTD. Your XML-based configuration files must conform to the rules set forth in these DTDs.

The XML document has to meet all the well-formedness constraints given in the XML specification. Specially for some special characters: ('), ("), (&), (<), and (>), they may appear in their literal form only when used as the following:

- Markup delimiters
- Within a comment
- A processing instruction

- A CDATA section

If they are needed elsewhere, they must be escaped using either numeric character references or the strings.

The apostrophe or single-quote character (') may be represented as “&apos;”, the double-quote character (") as "&quot;", the ampersand character (&) as "&amp;", the left angle bracket (<) as “&lt;”, and the right angle bracket (>) as “&gt;”.

Elements

Each DTD file consists of various *elements* that provide some function or task. These elements are contained within angled brackets (“< >”) and form the basis of the DTD. In the source code, each element is declared in the following way:

```
<!ELEMENT element_name>
```

In many cases, a given element will have subordinate elements, knows as *child elements*. The child elements associated with an element are contained within parentheses following the parent element’s declaration. For example:

```
<!ELEMENT element_name (child_element_name)>
```

If an element in the source code has a child element specified, the information for that child element can be found later in the code. An element can have more than one child element. Child elements that are only separated by a space can occur in any order within the parent element. For example:

```
<!ELEMENT element_name (child_element_name1 child_element_name2)>
```

If child elements are separated by a comma (“,”), then those child elements must appear in that order within the parent element in the code. For example:

```
<!ELEMENT element_name (child_element_name1, child_element_name2)>
```

If child elements are separated by a pipe (“|”), then only one of the child elements listed can be used. For example:

```
<!ELEMENT element_name (child_element_name1 | child_element_name2)>
```

In some cases, there can be restrictions or requirements placed on the usage of elements. Certain symbols placed immediately after an element name indicate these restrictions and requirements. For example:

- `<element_name>` (no symbol) — the element must appear just once.
- `<element_name?>` — the element can be omitted or can occur just once.
- `<element_name*>` — the element can be omitted or can appear one or more times.
- `<element_name+>` — the element must appear at least once, but can occur more than once as well.

Attributes

Element tags can include one or more optional or mandatory *attributes* that give further information about the elements they support. Attributes only can be specified within the element tag. The presence of an attribute within an element tag requires a corresponding value enclosed in quotes. For example:

```
<element_name attribute="value">
```

The type of value (a number, yes|no, a path) can vary depending on the type and nature of the attribute and element. The documentation for a given attribute will specify the types of values permitted.

An element can have any number of attributes. Multiple attributes in an element tag follow one after another, with no separators. For example:

```
<element_name attribute_name1="value" attribute_name2="value">
```

The attribute declaration always immediately follows its corresponding element declaration. In the source code of the OpenDeploy configuration DTD files, the attributes of a given element are declared in the following way:

```
<!ELEMENT Element_name>
<!ATTLIST Element_name
   attribute_name1        attribute1_type     attribute1_keyword
   attribute_name2        attribute2_type     attribute2_keyword
   >
```

The following sections describe attribute types and attribute keywords.

Attribute Type

The attribute type specifies the type of value that can be associated with the attribute. There are a variety of different attribute types possible. Some are immediately intuitive, such as (yes|no), where the choice is one of the values specified. Others require a more substantial explanation. Here is a list of commonly found attribute types within configuration DTDs:

- `CDATA` — character data such as a text string. CDATA is not recognized as markup language code.
- `ID` — an identifier for the element. No two elements can have the same ID attribute value in the same DTD. These types are usually unique identification numbers or strings.
- `IDREF` — a pointer to an ID reference. The value must match the value of an ID type attribute that is declared somewhere else in the same DTD.

There are other attribute types possible as well. Consult a book on XML for more information.

Attribute Keyword

The attribute keyword specifies action the server should take if an associated attribute is left out. One of the following values is used for the attribute keyword:

- `#REQUIRED` — the attribute is required and should be present. If it is missing, the configuration file is invalid and the application will not work.
- `#IMPLIED` — the attribute is not required. If the attribute has no value specified, the application will make a suitable substitution.
- `#FIXED` — the attribute value is fixed at a preset value. No modification to the attribute value is allowed.

Encoding

The encoding for the nodes configuration file can be encoding other than UTF-8. For example, if a value in the file contains Japanese characters, the encoding will need to be:

```
<? xml version="1.0" encoding="SHIFT_JIS" ?>
```

For French and German, the encoding value would be:

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
```

Check what the appropriate value is for any non-ASCII characters and modify the nodes configuration file encoding as needed. If no encoding is specified, UTF-8 will be used by default.

Naming Deployment Configurations

Deployment configuration file names can be any length up to the limit supported by the host operating system.

Deployment configurations residing on UNIX hosts cannot have spaces in the file names. Those running on Windows hosts may contain spaces.

All deployment configuration file names must have the `.xml` extension. The file cannot be read by OpenDeploy without this extension.

Deployment Configuration Structure

All OpenDeploy deployment configurations follow the same structure. All deployment configurations begin at the root element `deploymentConfiguration`. Within the `deploymentConfiguration` element are the following child elements:

- `localHost` (required) — specifies the deployment host name.
- `replicationFarmSet` (required) — the container element for target hosts and host groupings
- `definition` (required) — contains the source/target pairings, including the source and target file locations, and the type of deployment being performed
- `deployment` (required) — contains deployment features, including transactional and Deploy and Run
- `logRules` (optional) — specifies deployment logging settings. If no log rules are specified in the deployment configuration, OpenDeploy will reference the base server and receiver configuration files for logging instructions. If no logging information is present in those files either, OpenDeploy will use its default log settings. See Chapter 6, "Logging" on page 243 for more information.

The following example shows a simple deployment configuration using the minimum amount of information:

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentConfiguration>

    <localNode host="mars.mycompany.com"/>

    <replicationFarmSet>
        <replicationFarm name="MYFARMNAME">
            <nodeRef useNode="MyLocalHost"/>
        </replicationFarm>
    </replicationFarmSet>

    <definition name="MYDEFINITIONNAME">
        <source>
            <sourceFilesystem area="C:\Interwoven\OpenDeployNG\conf\dtd">
                <pathSpecification>
                    <path name="."/>
                </pathSpecification>
            </sourceFilesystem>
        </source>

        <target useReplicationFarm="MYFARMNAME">
            <comparisonRules  dateDifferent="yes"/>
            <permissionRules  file="0644" directory="0755"/>
            <targetFilesystem area="C:\Interwoven\OpenDeployNG\tmp"/>
        </target>
    </definition>

    <deployment transactional="no">
        <execDeploymentTask useDefinition="MYDEFINITIONNAME"/>
    </deployment>

    <logRules maxBytes="32Mb" level="verbose"/>

</deploymentConfiguration>
```

OpenDeploy will look to the base server (by default `odbase.xml`) or receiver (by default `odrcvr.xml`) configuration files, for direction if no configuration information for a particular feature or function is present in the deployment configuration. If there is no configuration information in these files either, OpenDeploy will either use its built-in settings, or ignore the feature.

Depending on the nature and complexity of the deployment, optional elements and attributes can be added as necessary. In the following example, a variety of additional elements and attributes have been added to the previous example. However, the basic structure remains the same:

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentConfiguration>

    <localNode host="mars.mycompany.com"/>

    <replicationFarmSet>
        <replicationFarm name="fan-out">
            <nodeRef useNode="a"/>
            <nodeRef useNode="b">
                <nextDeployment deployment="tier2" invokeOnSuccess="yes"/>
            </nodeRef>
            <nodeRef useNode="c"/>
        </replicationFarm>
    </replicationFarmSet>

    <definition name="enable_production">
        <source>
            <sourceFilesystem area="C:\Interwoven\OpenDeployNG\conf\dtd"">
                <pathSpecification>
                    <path name="."/>
                    <filters>
                        <excludePath subPath="out/exes"/>
                        <excludePattern regex="\.obj$"/>
                    </filters>
                </pathSpecification>
            </sourceFilesystem>
        </source>

        <target useReplicationFarm="fan-out">
            <comparisonRules dateDifferent="yes"/>
            <transferRules   doDeletes="yes"/>
            <permissionRules file="0644"
                             directory="0755"
                             user="webmaster"
                             group="webgroup"
                             />
            <targetFilesystem area="C:\Interwoven\OpenDeployNG\tmp"/>
        </target>
    </definition>
    <deployment transactional="no">
        <execDeploymentTask useDefinition="enable_production">
            <deployNRun>
                <dnrFile location="target" when="before" state="success"
                         mask="\.html$">
                    <script cmd="C:\bin\email_admin.bat -user rroe@mycompany.com"
                            as="webmaster"
                            where="C:\temp"
                            async="no"
                            />
                </dnrFile>
                <dnrDir location="target" when="after" state="failure"
                        mask="images">
                    <script cmd="C:\bin\email_admin.bat -user rroe@mycompany.com"
                            as="webmaster"
                            where="C:\temp"
                            async="no"
                            />
                </dnrDir>
                <dnrDeployment location="source" when="after" state="success">
                    <script cmd="C:\bin\email_admin.bat -user jdoe@mycompany.com"
                            as="webmaster"
                            where="C:\temp"
                            async="yes"
                            />
                </dnrDeployment>
            </deployNRun>
        </execDeploymentTask>
    </deployment>

    <logRules maxBytes="32Mb" level="verbose"/>

</deploymentConfiguration>
```

Specifying the Deployment Host

The deployment host is the OpenDeploy server that will serve as the source host of the deployment. The deployment configuration must reside on the host that will start the deployment. The deployment configuration cannot exist remotely from the source host. The deployment host is specified as the `localNode` element's `host` attribute value. For example:

```
<deploymentConfiguration>
   <localNode host="mars.mycompany.com" />
   ...
</deploymentConfiguration>
```

This value must be the OpenDeploy host's fully qualified DNS host name or IP address. It cannot be the logical name of the host. This host must also be listed in the host's nodes configuration file (by default `odnodes.xml`).

In some cases you might compose or modify a deployment configuration on your local computer, but intend to move it to an OpenDeploy base server host for actual usage. In these cases, the `host` attribute value must be that of the OpenDeploy base server host.

Target Replication Farms

Target replication farms are groupings of OpenDeploy target host nodes under a single named element. All target hosts listed in a replication farm will receive the same set of deployed files, unless any overrides are specified. Replication farms allow you to simply the deployment process by deploying a single set of files to a large number of targets without having to individually configure each one.

Each deployment configuration must have at least one target replication farm, even if that farm consists of only a single target host node. You can have as many additional replication farms as you want, as long as each one is uniquely named. An individual target host node can belong to more than one replication farm.

Each target replication farm is designated by the presence of a `replicationFarm` element.The `replicationFarm` element's `name` attribute must contain a unique name. Within each `replicationFarm` element are `nodeRef` child elements. You must have a corresponding `nodeRef` element for each target host node to which you want to deploy files. Each `nodeRef` element must also correspond to a node entry in the nodes configuration file (by default `odnodes.xml`). The `nodeRef` element's `useNode` attribute value must be the same as the target host node's logical name (specified as the node element's name attribute value in the nodes configuration file).

Each individual `replicationFarm` element is contained within a single `replicationFarmSet` element for the deployment configuration. In the following example, the target replication farms *europe* and *asia* are specified within the deployment configuration:

```
<deploymentConfiguration>
   <localNode host="mars.mycompany.com" />
   <replicationFarmSet>
      <replicationFarm name="europe">
         ...
      </replicationFarm>
      <replicationFarm name="asia">
         ...
      </replicationFarm>
   </replicationFarmSet>
   ...
</deploymentConfiguration>
```

Within the target replication farm *europe*, the individual target host nodes making up the replication farm are listed, for example:

```
<replicationFarm name="europe">
   <nodeRef useNode="england" />
   <nodeRef useNode="france" />
   <nodeRef useNode="spain" />
</replicationFarm>
```

These target host nodes also must be listed in the nodes configuration file for the deployment host, for example:

```
<nodeSet name="od_receiver_nodes">
   <node name="england"    host="london.mycompany.com"      port="20014" />
   <node name="france"     host="paris.mycompany.com"       port="20014" />
   <node name="spain"      host="madrid.mycompany.com"      port="20014" />
</nodeSet>
```

Definitions

A *definition* is a uniquely-named pairing of the source file location on the sending OpenDeploy host with one or more target file locations on the receiving hosts. The target file location specified in the definition is then applied to the target hosts listed within the replication farm to determine which targets receive the deployed files, and where on the target host's file system those files will reside.

A definition is specified in the deployment configuration with the `definition` element. Its unique name is the value of the `name` attribute. Within the `definition` element are the `source` and `target` child elements. For example:

```
<deploymentConfiguration>
   ...
   <definition name="webfiles">
      <source>
         ...
      </source>
      <target ...>
         ...
      </target>
   </definition>
   ...
</deploymentConfiguration>
```

Each deployment configuration must have at least one definition. You can have as many additional definitions as you want, as long as each one is uniquely named.

Source File Location

The source file location is where the source files participating in a deployment reside. This location can either be a file system location or a TeamSite area (if TeamSite is also installed on your OpenDeploy source host). If the deployment is comparison-based, the source file location participates in the file comparison, either with a file system location on the target host, or with another TeamSite area on the source host. If the deployment is file list-based, those files that are deployed must reside in the source file location.

The source file location is defined within the definition by the `source` element, and its `sourceFilesystem` (for directory comparison and file list deployments) and `sourceTeamsite` (for TeamSite comparison deployments) child elements. For example:

```
<definition name="webfiles">
   <source>
      <sourceFilesystem area="C:\website\files" >
         ...
      </sourceFilesystem>
   </source>
   ...
</definition>
```

You can have multiple source file locations configured within the definition. For example:

```
<definition name="webfiles">
   <source>
      <sourceFilesystem area="C:\website\files" >
         ...
      </sourceFilesystem>

      <sourceFilesystem area="C:\images" >
         ...
      </sourceFilesystem>
   </source>
   ...
</definition>
```

However, you should take precautions to avoid overwriting the deployed files from one source file location with those of another.

All your source file locations within a definition must be either a filesystem location or TeamSite area. You cannot combine both types within a single definition.

Target File Location

The target file location is defined within the definition by the `target` element. Only a single target file location is allowed in each definition.

The `useReplicationFarm` attribute specifies the target replication farm on whose target host nodes the target file location will receive the deployed files. The target replication farm value must match an entry present in the source host's nodes configuration file (by default `odnodes.xml`). See "Target Replication Farms" on page 116 for more information.

The target file location must be a file system location. The target hosts are not listed in the definition, only the target file location.

```
<definition name="webfiles">
   ...
   <target useReplicationFarm="MYFARMNAME">
      <targetFilesystem area="D:\website\files" />
   </target>
</definition>
```

TeamSite areas are not supported in the target file location. However, if you want to deploy files to a TeamSite workarea, you can use the TeamSite area virtual path as the value for the target file location. For example:

```
<targetFilesystem area="Y:\default\main\dev\WORKAREA\jdoe">
```

Deploying to Mixed Platform Targets

Any target file location listed in a definition is assumed by OpenDeploy to apply to all the target hosts listed in the target replication farms. Deployments to UNIX host will only work if you use forward slash ("/") as path delimiter. Deployments to Windows systems will work with either forward ("/") or backward slash ("\") as path delimiter, since OpenDeploy converts forward slashes to back slashes on Windows.

File Filters and Rules

Also included in a definition are all the rules and filters that determine what files can be deployed by the sending host and received by the target hosts. These include the following:

- File path exclusion filters
- File name pattern exclusion filters
- File comparison rules
- File transfer rules
- File permission rules
- Transfer rules for symbolic links
- Target override rules

See Chapter 4, “Deployment Features” for more information on these features, and how to configure them in your deployment.

Deployment Tasks

The deployment element contains attributes and child elements that allow you to configure the following items:

- Transactional functionality
- Definition-specific configurations for down-rev deployments and Deploy and Run scripting

Transactional

The deployment element provides the ability to make the deployment configuration transactional. If a deployment with the transactional feature enabled fails to successfully deploy all the appropriate files to the target hosts, OpenDeploy will roll back the deployment and restore the target host's file locations to their previous state. See "Transactional Deployments" on page 144 for more information.

You can enable this feature by assigning a value of yes to the transactional attribute, for example:

```
<deploymentConfiguration>
   ...
   <deployment transactional="yes">
      ...
   </deployment>
</deploymentConfiguration>
```

Definition-Specific Configurations

Within the deployment element you can configure certain functionality on a definition-specific basis. You can specify a particular definition through the execDeploymentTask element. Each deployment element must have at least one execDeploymentTask child element. If there is only one definition in the deployment configuration, the execDeploymentTask element's useDefinition attribute value must by that definition's name. For example:

```
<deployment transactional="yes">
   <execDeploymentTask useDefinition="europe" />
</deployment>
```

If you have multiple definitions in a deployment configuration, then you can specify one, some, or all of those definitions with a separate instance of the `execDeploymentTask` element. For example:

```
<deployment transactional="yes">
   <execDeploymentTask useDefinition="europe">
      ...
   </execDeploymentTask>
   <execDeploymentTask useDefinition="asia">
      ...
   </execDeploymentTask>
</deployment>
```

Within the `execDeploymentTask` element, you can configure the following features:

- Downward revision deployments
- Deploy and Run scripts

Deploy and Run

Deploy and Run allows you to configure OpenDeploy to execute an external script at a specified stage of the deployment.This stage can be the deployment of a particular type or class of file or directory, or even the success of the deployment itself. See “Deploy and Run” on page 201 for more information.

Deploy and Run scripting is assigned on a definition-specific basis within the `execDeploymentTask` element. The `deployNRun` child element is the container for Deploy and Run configuration. For example:

```
<execDeploymentTask useDefinition="europe">
   <deployNRun>
      ...
   </deployNRun>
</execDeploymentTask>
```

Directory Comparison Deployments

During a *directory comparison*, OpenDeploy compares the directories and files on the source host with another set of files and directories residing on the target host. The files and directories that meet the deployment criteria as a result of this comparison can then be moved to the target host.

In Figure 28, the source host *mars* has a file system area defined as:

```
C:\dev\website\files
```

The target host *venus* has a file system area defined as:

```
D:\website\files
```

The deployment configuration file specifies the rules for determining which files should be deployed. It also specifies any additional rules or actions that are applied to the deployed files. OpenDeploy uses these rules during the comparison and deployment phases of its operation. See “Deployment Types” on page 36 for more information.

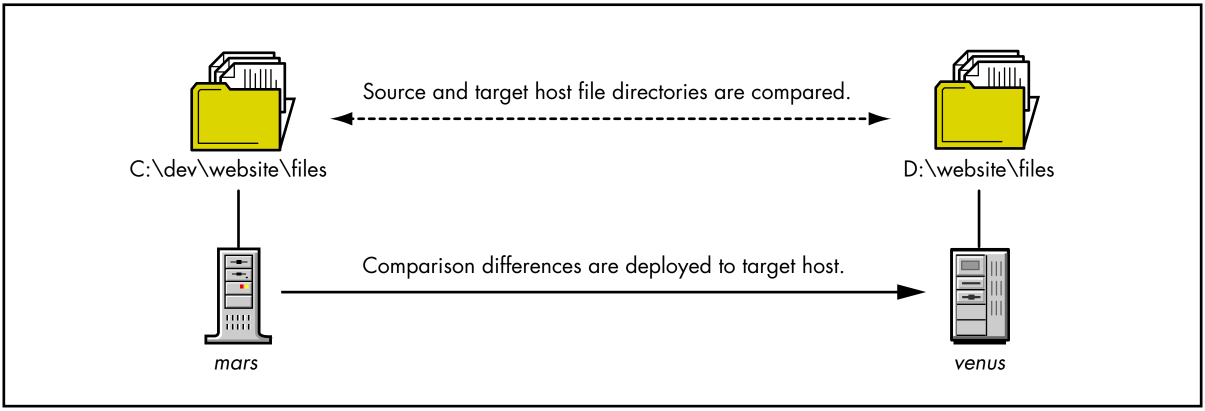

Source and target host file directories are compared.
C:\dev\website\files
D:\website\files
Comparison differences are deployed to target host.
mars
venus

Figure 28: Directory Comparison

Defining the Source Host File Area

Directory comparison deployments are determined by the `sourceFilesystem` element. This element contains the `area` attribute where the directory containing the source host's files is specified. The `area` attribute value specifies the absolute path for a file system containing the host files. For source hosts, this indicates the path where the files currently reside. For example:

`area="/website/files"` *or*

`area="C:\website\files"`

Specifying TeamSite Areas

You can specify any TeamSite area as the source file location by assigning its file system path equivalent as the value of the `area` attribute. For example, the Windows file system path equivalent of the following TeamSite path:

```
/default/main/dev/EDITION/ed01012001
```

might be reflected in the deployment configuration as:

```
<sourceFilesystem area="Y:\default\main\dev\EDITION\ed01012001">
```

OpenDeploy also permits the use of TeamSite area paths ending in `EDITION` and `IW_PREV` as the source file location in directory comparison deployment configurations. For example:

`<sourceFilesystem area="/default/main/dev/EDITION">` *or*

`<sourceFilesystem area="/default/main/dev/EDITION/IW_PREV">`

The use of the values `EDITION` and `IW_PREV` refers to the most recent and next-to-most recent published TeamSite editions. By using the TeamSite paths for `EDITION` and `IW_PREV`, you can automatically specify the current or next-to-most current editions without having to modify the deployment configuration.

You can also include `//IWSERVER` at the beginning of your TeamSite area path. This component defaults the source file area to different root file system locations for depending on the host's operating system. See "Using //IWSERVER in the TeamSite Path" on page 132 for more information.

In a directory comparison deployment configuration, the edition specified by the use of EDITION or IW_PREV would still be compared to the target file location, and the differences deployed. In the following example, a directory comparison deployment uses the TeamSite path EDITION for the source file location:

```
<source>
   <sourceFilesystem area="/default/main/dev/EDITION">
      ...
   </sourceFilesystem>
</source>

<target useReplicationFarm="MYFARMNAME">
   <targetFilesystem area="D:\website\files" />
</target>
```

Specifying a Location Within the Source File System Area

There may be times when you want to identify a particular location within the specified area for a deployment. You can specify locations within the source area by configuring additional elements and attributes within the source element.

The pathSpecification and path elements allow you to specify a particular relative location within the designated source area. In the following example:

```
<source>
   <sourceFilesystem area="C:\website\files">
      <pathSpecification>
         <path name="monthly" />
      </pathSpecification>
   </sourceFilesystem>
</source>
```

the file system area is specified as C:\website\files. In order to further specify the subdirectory monthly within the area, the path element's name attribute value was specified as monthly.

The name attribute is the directory relative to the area specified in the sourceFilesystem attribute for your source host. If you want to specify a location multiple levels below the area, you can enter the path to that location relative to the area. For example, if you want to specify the source location monthly\january relative to the area, the path element would be:

```
<path name="monthly\january" />
```

The pathSpecification and path elements are required in the deployment configuration file. However, if you do not want to specify a source host location any narrower than the specified area, you can simply list the path element's name attribute as ".", which indicates the area location. This is the default setting for all the OpenDeploy sample configuration files. The following example shows a simple source element where no source host location is specified beyond the area:

```
<source>
   <sourceFilesystem area="/website/files">
      <pathSpecification>
         <path name="." />
      </pathSpecification>
   </sourceFilesystem>
</source>
```

Defining the Target File Location

The target file location for a directory comparison deployment is specified by the targetFilesystem element. This element and its area attribute specify the file system location where the target files will reside after the deployment. Like the sourceFilesystem element, the targetFilesystem element contains an area attribute where you must specify an absolute path to the target host file location. For example:

```
<targetFilesystem area="D:\website\files" />
```

After the sourceFilesystem and targetFilesystem elements are defined, the directory comparison can take place. The following example below shows the pairing between the source and target host file system-based locations:

```
<source>
   <sourceFilesystem area="/website/files">
      ...
   </sourceFilesystem>
</source>

<target useReplicationFarm="MYFARMNAME">
   <targetFilesystem area="D:\website\files" />
</target>
```

You can specify a UNIX path for the area attribute value and have it applied to a Windows target. See "Deploying to Mixed Platform Targets" on page 120 for more information.

Root as Target Host Area

You can specify the root directory of a Windows filesystem (`area="C:\"`) as a target host file area of the deployment. Deployments to the root directory of a UNIX target host ("/") are not supported.

TeamSite Comparison Deployments

TeamSite comparison deployments are specified by the `sourceTeamsite` element and its attributes. The `area` attribute's value is the path of the TeamSite area containing the source host's files. In a TeamSite comparison, the source host files are located in a TeamSite area and are compared with a second TeamSite area in the same backing store on a host with TeamSite installed. The differences between the two TeamSite areas are what is deployed to the target host. You can configure OpenDeploy to compare two TeamSite areas in any single backing store on a multi-backing store TeamSite host.

This type of comparison is most effective if the second TeamSite area contains a mirror image of the files on the eventual target host. This differs from a directory comparison deployment, where files on the source host are compared with a corresponding file location on the target host.

In Figure 29, the source host *mars* is a running TeamSite software in addition to OpenDeploy.

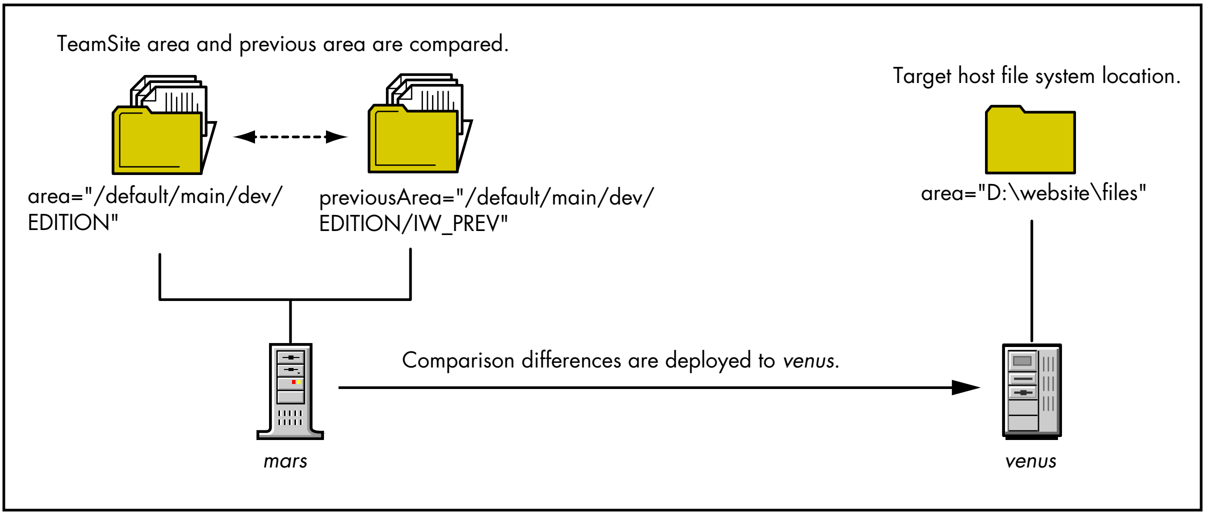

TeamSite area and previous area are compared.
Target host file system location.
area="/default/main/dev/
EDITION"
previousArea="/default/main/dev/
EDITION/IW_PREV"
area="D:\website\files"
Comparison differences are deployed to *venus*.
mars
venus

Figure 29: TeamSite Comparison Deployment

It has the most recent TeamSite edition as the value for the `area` attribute:

```
/default/main/dev/EDITION
```

This area is known as the *primary area* and its TeamSite location is the value for the `area` attribute.

The previous version of this edition represents a mirror image of the files residing on the eventual target host *venus*:

```
/default/main/dev/EDITION/IW_PREV
```

This previous version is known as the *previous area* and its TeamSite location is the value for the `previousArea` attribute. OpenDeploy compares the two TeamSite areas specified in the `area` and `previousArea` attributes, and determines which files in the more recent edition need to be deployed to the target host. See "TeamSite Comparison" on page 39 for more information on TeamSite comparison-based deployments.

Defining the Source Host TeamSite Areas

The areas comprising a TeamSite comparison deployment are specified by the `sourceTeamsite` element. The `sourceTeamsite` element contains two attributes:

- `area` — defines the primary TeamSite area
- `previousArea` — defines the previous TeamSite area

Specifying the Primary TeamSite Area

The `area` attribute defines the TeamSite host area where the new files to be compared are located. This area is typically an edition or a staging area. For example:

```
/default/main/dev/STAGING
```

Although you can deploy files from a workarea, it is not recommended, because files in a workarea do not have file versioning and other TeamSite content management features applied to them. It is preferred to submit files from a workarea to the staging area, and to perform the deployment from there.

To designate this attribute as the most recently published TeamSite edition, enter the TeamSite path ending simply with “EDITION.” OpenDeploy will automatically access the most recently published edition path. For example:

```
/default/main/dev/EDITION
```

To specify a another edition for the `area` attribute, enter the complete TeamSite path to that edition. For example:

```
/default/main/dev/EDITION/ed01022001
```

Specifying the Previous Area for Comparison

The `previousArea` attribute defines the TeamSite host area against which the content in the `area` attribute is compared. The content of the `previousArea` should be a mirror image of the content contained on the target host receiving the deployed files.

To designate this attribute as the edition that immediately preceded the most recently published one (as specified by the use of `EDITION` in the `area` value), enter the TeamSite path ending simply with `EDITION/IW_PREV`. OpenDeploy will automatically access the next-to-most recently published edition path. For example:

```
/default/main/dev/EDITION/IW_PREV
```

To specify another edition for the `previousArea` attribute, enter the complete TeamSite path to that edition. For example:

```
/default/main/dev/EDITION/ed01012001
```

The two designated TeamSite areas within the `sourceTeamsite` element are compared during the TeamSite comparison, and the files that meet the deployment criteria are deployed to the target host.

In the following example:

```
<sourceTeamsite
   area="/default/main/dev/EDITION"
   previousArea="/default/main/dev/EDITION/IW_PREV"
   >
   ...
</sourceTeamsite>
```

the last two TeamSite editions are compared with each other, and the differences deployed to the target host.

In some cases, you might want to deploy the entire contents of a TeamSite area. You can perform this task by specifying a TeamSite area that contains no files as the value for the previousArea attribute, such as the initial edition that is generated for the TeamSite base branch (EDITION/INITIAL). For example:

```
<sourceTeamsite
   area="/default/main/dev/EDITION"
   previousArea="/default/main/EDITION/INITIAL"
   >
   <pathSpecification>
      <path name="." />
   </pathSpecification>
</sourceTeamsite>
```

In these types of deployments, the path element's name attribute value can only be ".” No other locations can be specified for the name attribute.

The next section further describes the use of the path and pathSpecification elements.

Using //IWSERVER in the TeamSite Path

You can include `//IWSERVER` at the beginning your TeamSite path for the `area` and `previousArea` attribute values, for example:

```
area="//IWSERVER/default/main/dev/EDITION" or

previousArea="//IWSERVER/default/main/EDITION/INITIAL"
```

Inclusion of `//IWSERVER` in the TeamSite path translates into the following default source file location, depending on the operating system:

- Windows — `Y:\default`
- UNIX — `/iwmnt/default`

For example, if you specify the following on a deployment configuration:

```
area="//IWSERVER/default/main/dev/EDITION"
```

On a Windows host, the file system equivalent is:

```
area="Y:\default\main\dev\EDITION"
```

while on a UNIX host it is:

```
area="/iwmnt/default/default/main/dev/EDITION"
```

Using the `//IWSERVER` gives you a greater level of flexibility in running the same deployment on both Windows and UNIX hosts, since you do not have to change the source file location paths to reflect the host operating system.

Specifying a Location Within the Source TeamSite Area

You can specify narrower locations within TeamSite areas in the same manner as you can with file system-based areas. The elements and attributes for specifying a location within a TeamSite area-based source host area are essentially the same as for a file system-based source host. The location is relative to the area specified in the `sourceTeamsite` element. However, in a TeamSite comparison deployment, the source host area is specified as a TeamSite area.

You can use the pathSpecification and path elements and their attributes to specify a location within both TeamSite areas being compared (as indicated by the values of the area and previousArea attributes), just as you did with file system-based deployment source hosts. The relative directory or path you specify in the path element's name attribute applies equally to the area and previousArea locations. You cannot specify a narrowed location on only one of the two areas.

In the following example:

```
<source>
   <sourceTeamsite
      area="/default/main/dev/EDITION"
      previousArea="/default/main/dev/EDITION/IW_PREV"
      >
      <pathSpecification>
         <path name="monthly" />
      </pathSpecification>
   </sourceTeamsite>
</source>
```

the two TeamSite areas being compared are the current edition:

```
/default/main/dev/EDITION
```

and the previous edition:

```
/default/main/dev/EDITION/IW_PREV
```

By specifying the value monthly for the path element's name attribute, the two TeamSite areas being compared are narrowed to the directory monthly located in each TeamSite area.

The two TeamSite areas being compared are now effectively the following:

area — /default/main/dev/EDITION/monthly *and*

previous area — /default/main/dev/EDITION/IW_PREV/monthly

If you are deploying the entire contents of the source TeamSite area by specifying the previousArea attribute location as an empty TeamSite area (such as the initial edition that is generated for the TeamSite base branch (EDITION/INITIAL)), the path element's name attribute value can only be "." No other locations can be specified. For example:

```
<sourceTeamsite
   area="/default/main/dev/EDITION"
   previousArea="/default/main/EDITION/INITIAL"
   >
      <pathSpecification>
         <path name="." />
      </pathSpecification>
</sourceTeamsite>
```

Defining the Target File Location

Unlike a directory comparison where the target host is an integral part of the comparison, in a TeamSite comparison the target host simply receives the files the source host determines are appropriate. The target file location for a TeamSite comparison deployment can only be a file system location.

Use the `targetFilesystem` element to designate this type of target file location. For example:

`<targetFilesystem area="C:\website\files">` *or*

```
<targetFilesystem area="/website/files">
```

To deploy files to a TeamSite workarea on your target host, use the file system location rather than the TeamSite path in your configuration. For example:

```
<targetFilesystem area="Y:\default\main\dev\WORKAREA\jdoe">
```

You can specify a UNIX path for the `area` attribute value and have it applied to a Windows target. See “Deploying to Mixed Platform Targets” on page 120 for more information.

Use With Deploy and Run Scripts

Evaluation of the `area` and `previousArea` attribute values in TeamSite comparison deployments with respect to the latest and next-to-latest editions, occurs *before* the running of any Deploy and Run scripts. See “Deployment-Based” on page 206 for more information.

Redeploying Legacy Web Sites

If you are using OpenDeploy in conjunction with TeamSite, you can make a copy of a set of deployed files and store it for later use. This feature is handy if you need to "roll back" an existing Web site to a previous version, or recreate a Web site as it once looked. When a set of Web files on a TeamSite server is complete and ready to deploy, make a TeamSite edition of those files. Include the date or some other identifying element in the edition name. Later, if you need to recreate a legacy Web site, you can deploy that saved edition.

You can deploy the TeamSite edition using the TeamSite comparison method. Specify the `area` attribute value as the path to the TeamSite edition you want to deploy. Specify a TeamSite area that contains no value for the `previousArea` attribute, such as the initial edition that is generated for the TeamSite base branch. When the deployment is run, all the files in the TeamSite edition you need to restore your legacy Web site will be redeployed to the target host. See "TeamSite Comparison Deployments" on page 128 for more information.

File List Deployments

File list deployments do not compare source or target files, either in a file system location or a TeamSite area. Instead, OpenDeploy simply moves the files from the source host to the target host based on the files and directories indicated in the file list itself. Only one list of files per deployment is allowed. All deployment criteria and functionality can be applied to file list deployments.

Unlike directory comparison and TeamSite comparison types of deployments, a file list deployment does not have its own unique element that defines the deployment. Instead, the file list deployment is indicated by the presence of the `filelist` attribute and its value in the `sourceFilesystem` element.

In Figure 30, the source host *mars* is performing a file list deployment to the target host *venus*.

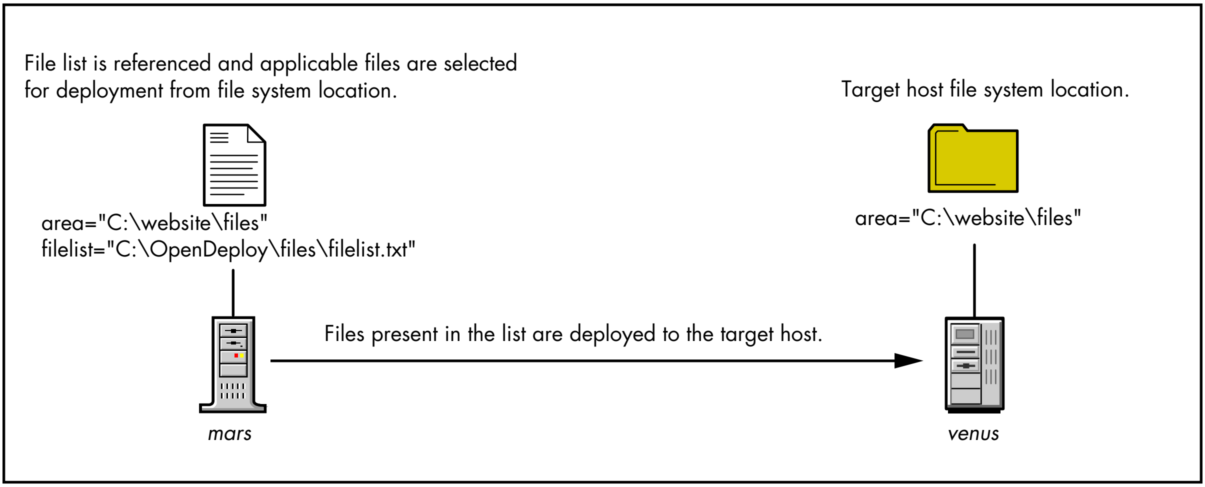

File list is referenced and applicable files are selected for deployment from file system location.
Target host file system location.
area="C:\website\files"
filelist="C:\OpenDeploy\files\filelist.txt"
area="C:\website\files"
Files present in the list are deployed to the target host.
mars
venus

Figure 30: File List Deployment

The source host references the file specified by the `filelist` attribute in the `sourceFilesystem` element:

```
C:\OpenDeploy\files\filelist.txt
```

and proceeds to deploy the files listed there to the target host. The value specified in the `area` attribute:

```
C:\website\files
```

defined as the location of the files to be deployed from the source host, is the same as in a directory comparison deployment.

Specifying the File List

File list deployments are specified in the `sourceFilesystem` element's `filelist` attribute. The `filelist` attribute specifies the absolute path to the file list file. For example:

```
<sourceFilesystem
   area="C:\website\files"
   filelist="C:\OpenDeploy\files\filelist.txt"
   >

   ...
</sourceFilesystem>
```

The `filelist` attribute and its value within the `sourceFilesystem` element must be present for OpenDeploy to recognize the configuration as a file list deployment. Without it, the deployment is performed as a directory comparison.

Editing the File List

The file list is a text file that you can create and modify using your favorite text editor. This list contains a series of entries of files and directories that are relative to the `area` attribute value specified in the `sourceFilesystem` element. Here is an example of file list entries:

```
www/index.html
www/andre/index.html
www/products.html
```

You must follow the path syntax of your source host computer type when editing the file list. Forward slashes ("/") can be used for either Windows or UNIX hosts:

```
www/andre/index.html
```

while backslashes ("\") are only permitted on Windows hosts:

```
www\andre\index.html
```

Because these files and directories are relative to the specified file system, you do not need to enter the absolute path of each entry in the file list. Instead, just enter the path relative to that `area` attribute you specified in the `sourceFilesystem`. For example, if you had specified the `sourceFilesystem` element's `area` attribute as having the following value:

```
<sourceFilesystem
   area="C:\website\files"
   filelist="C:\OpenDeploy\files\filelist.txt"
   >
   ...
</sourceFilesystem>
```

then the entries in the file list would effectively have the following absolute path:

```
C:\website\files\www\index.html
C:\website\files\www\andre\index.html
C:\website\files\www\products.html
```

File List Deployments from TeamSite Areas

In some cases, you might want a file list deployment to originate from a TeamSite area on the source host. Because the `filelist` attribute is only associated with the `sourceFilesystem` element, you cannot specify a TeamSite area. However, you can specify the file system path to that TeamSite area on the source host. For example, if you wanted to designate the following TeamSite area as the source area:

```
/default/main/dev/EDITION
```

you would enter the file system location equivalent of the TeamSite area in the `sourceFilesystem` element's `area` attribute. For example:

`area="/iwmnt/default/main/dev/EDITION/ed01012001"` *or*

`area="Y:\default\main\dev\EDITION\ed01012001"`

Unlike TeamSite area-based areas, where you can simply state the value "EDITION" to indicate the latest TeamSite edition available, when entering a file system-based path to a TeamSite area, you must enter the entire path to the specific edition you want, even if it is the most recent one.

Defining the Target Host Location

The same principles apply to the target host area for file list deployments as for directory or TeamSite comparison deployments. The target host area must be a file system location. However, unlike in a directory comparison deployment, the target host area in a file list deployment is not being compared to the files in the source host area. Rather, the target host area is simply a location where the files deployed as specified in the file list end up. See "Defining the Target File Location" on page 127 for general configuration information, including specifying a location within the file system-based target area.

Using doDeletes with File List Deployments

File list deployments support the use of the `doDeletes` option. Those files that you want to be deleted at the target must be named in the deployment file list file, and must not be present in the source file directory of the source host. See "Target Host File Deletions Using Filtered Deployments" on page 184 for more information on the use of the `doDeletes` option.

Defining Source-Based Overrides

You can override global target-side attributes, such as the target file location and certain rules and filters with alternate attributes on a source-level basis. This feature allows files in a specified source file location to be deployed to an alternate target file location, and also overrides any existing target-side rules or filters. You can also use this feature to deploy a single set of files to multiple target file locations within the same deployment configuration.

One use of source-based overrides is when there are metadata files associated with a set of deployed files. You might want to also deploy those metadata files, but to a different target file location. By specifying this alternate target file location in conjunction with the metadata source file location, you can deploy both the main files and the metadata files within the same definition.

Source-based overrides are specified in the `targetRules` element. When this element is present in the deployment configuration, OpenDeploy will override the target file location in the `targetFilesystem` element and both compare (if necessary) and send the files to the path location specified as the `targetRules` element's `area` attribute value instead.

The targetRules element resides within the pathSpecification element associated with each occurrence of the sourceFilesystem or sourceTeamsite elements in a deployment configuration. Each sourceFilesystem or sourceTeamsite element can have one occurrence of the targetRules element within each occurrence of the pathSpecification element.

The following example illustrates how a source file location can compare and deploy files to an alternate target file location (D:\metadata\files) at the same time another source file location deploys to the standard target file location (D:\website\files), all within the same definition:

```
<source>
   <sourceFilesystem area="C:\website\files">
      ...
   </sourceFilesystem>

   <sourceFilesystem area="C:\metadata\files">
      <pathSpecification>
         ...
         <targetRules area="D:\metadata\files" />
      </pathSpecification>
   </sourceFilesystem>
</source>

<target useReplicationFarm="MYFARMNAME">
   <targetFilesystem area="D:\website\files">
</target>
```

You can specify source-based overrides for the following OpenDeploy features:

- Filters
- Transfer rules
- Comparison rules
- Permission rules

See Chapter 4, “Deployment Features” for more information on these features.

Defining Target-Based Overrides

There are a variety of situations where the single set of values specified in the `target` element of a deployment configuration cannot apply to all the target hosts in a multi-target deployment equally or successfully. These situations include:

- A different target file location is required for one or more target hosts.
- Features need to be added or modified on a target-specific basis.

Specifying Different Target Areas

You can also use this feature to specify different target host areas even when the source and target hosts are all of the same platform. In the following example, the target hosts *mars*, *jupiter*, and *saturn* are all Windows servers.

```
<replicationFarmSet>
   <replicationFarm name="MYFARMNAME">
       <nodeRef useNode="mars" />
       <nodeRef useNode="jupiter">
          <targetRules area="c:\website\western"/>
       </nodeRef>
       <nodeRef useNode="saturn">
          <targetRules area="c:\website\eastern"/>
       </nodeRef>
   </replicationFarm>
</replicationFarmSet>
```

Using the same `targetFilesystem` element and `area` attribute values from before, *mars* accepts that area location to receive the deployed files, while *jupiter* and *saturn* each override that location with ones unique to their own file systems.

Defining Features on a Target-Specific Basis

You can specify certain features to apply to particular target hosts. A feature defined within the `nodeRef` element for a target host overrides any comparable feature or attribute value defined within the `target` element of the deployment configuration, as well as any source-based overrides specified within the `pathSpecification` element. Within each `nodeRef` element you can add additional elements to specify features for that target host including:

- Filters
- Transfer rules
- Comparison rules
- Permission rules

See Chapter 4, "Deployment Features" for information on these features.

Deployment Logging

Logging settings in a deployment configuration affect the following types of log files:

- Source macro log
- Source micro log
- Receiver macro log
- Receiver micro log

Base server and receiver log files are not affected by logging settings present in deployment configurations.

You can specify the following logging rules in a deployment configuration:

- Rollover threshold size for a deployment's micro and macro log files
- Logging level

One or both of these settings will override any equivalent settings present in the base server or receiver configurations as they apply to deployment logs. Logging level settings in a deployment configuration can be overridden only when a different level is specified when a deployment is started manually in the OpenDeploy user interface, or by using the `iwodstart` command.

You cannot specify a different log file location in a deployment configuration. That functionality only exists in the base server or receiver configurations files.

See Chapter 6, "Logging" on page 243 for a complete description of how OpenDeploy logs deployments, and on how to manage your log files.

Transactional Deployments

Transactional deployments give you an added level of security for maintaining the integrity of your Web sites during deployments by automatically rolling back a deployment and restoring the target host files to their previous states in the event one or more target deployments fail.

Transactional deployments are particularly useful where a number of recipient target hosts must have their Web sites synchronized with each other. If one or more of the deployments to these target hosts fail, it may be preferred to restore some or all of them (even the target hosts whose deployments succeeded) back to their previous states until another deployment can be attempted.

In Figure 31, the source host *mars* attempts a transactional deployment to the target host *venus*. The deployment can be of any type. During the deployment, file transmission to the target host is interrupted halfway through the process and is considered to be a failed deployment. After OpenDeploy becomes aware that this transactional deployment has failed, it restores *venus*' file area containing the deployed files back to its original state.

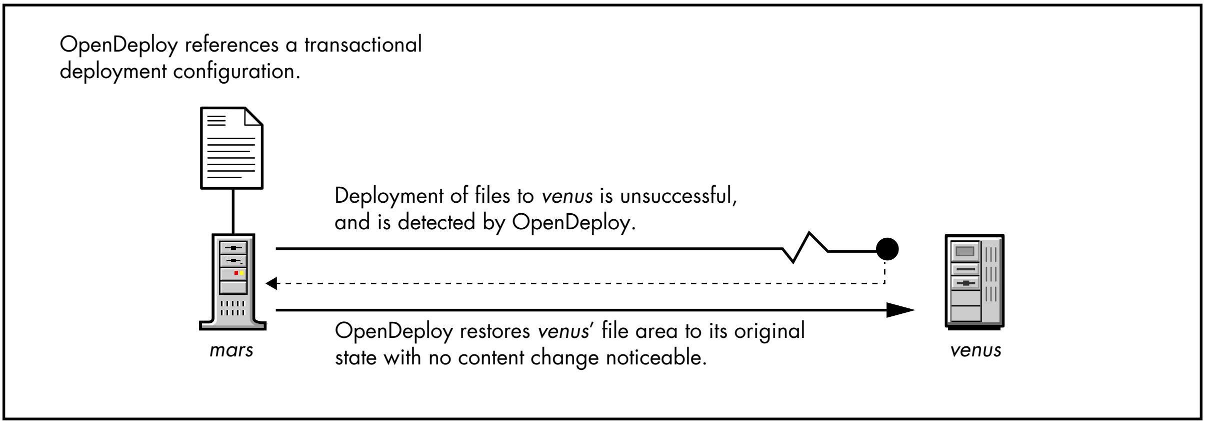

OpenDeploy references a transactional deployment configuration.
Deployment of files to *venus* is unsuccessful, and is detected by OpenDeploy.
OpenDeploy restores *venus*' file area to its original state with no content change noticeable.
mars
venus

Figure 31: Transactional Deployment

Transactional deployments are enabled only in the deployment configuration file an attribute of the `deployment` element. Give the value `yes` if you want the deployment to be transactional. For example:

```
<deployment transactional="yes">
```

Fan-out deployments can use the transactional feature to roll back a deployment and restore files on multiple targets. Similarly, multi-tiered deployments can use the transactional feature to roll back deployments across tiers. See "Transactional Targets in Fan-Out Deployments" on page 147 and "Transactional Targets in Multi-Tiered Deployments" on page 153 for more information.

Fan-Out Deployments

OpenDeploy can deploy the same set of files to multiple target hosts as part of a single deployment. Deploying to multiple targets in this way is called a *fan-out deployment*. Fan-out deployments are often preferred if your organization has several production Web servers, each of which must have its Web content synchronized with the others. A fan-out deployment automatically deploys to all target hosts simultaneously with no more effort on your part than if you were deploying to a single target host.

The same deployment configuration used to deploy files to a single target host can be modified to include all targets. Any type of deployment can be used, and all deployment rules and features are allowed in fan-out deployments. Although deployment to each target is identical by default, you can also modify the fan-out deployment configuration to allow exemptions to the rules on a target-specific basis.

In Figure 32, the source host *mars* performs a fan-out deployment to three mirrored production servers: *venus*, *jupiter*, and *mercury*.

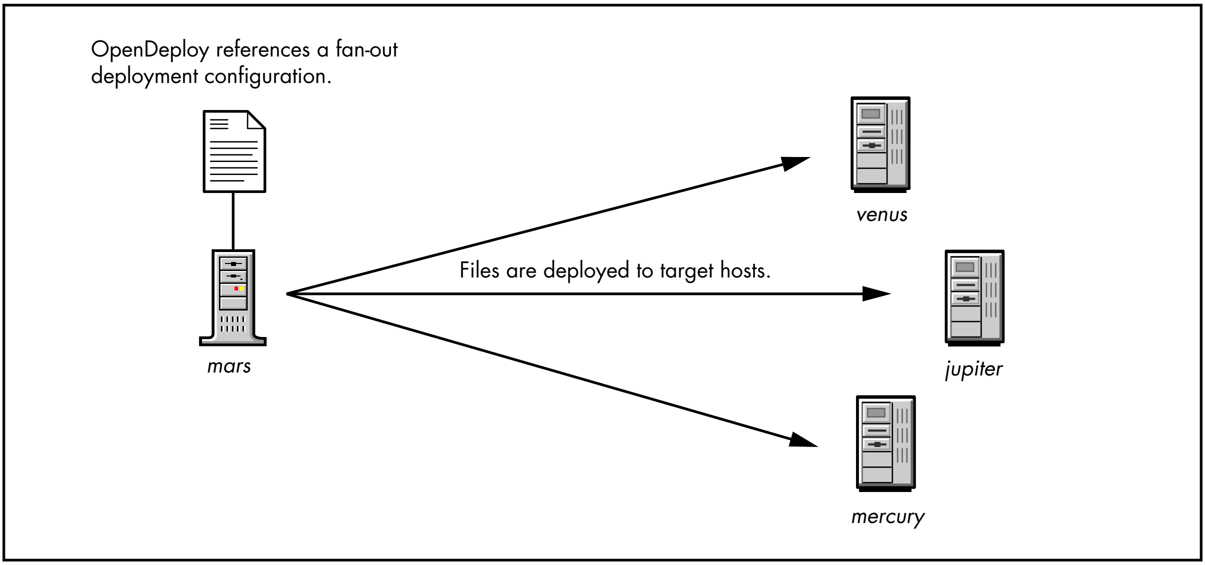

OpenDeploy references a fan-out deployment configuration.
Files are deployed to target hosts.
mars
venus
jupiter
mercury

Figure 32: Fan-Out Deployment

You configure fan-out deployments using one of the following methods:

- Use multiple `nodeRef` elements within the `replicationFarm` element:

  ```
  <replicationFarm ...>
     <nodeRef useNode="venus" />
     <nodeRef useNode="jupiter" />
  </replicationFarm>
  ```

  The `useNode` attribute value in a `nodeRef` is the logical name for a specific target host. That logical name must appear in the source host's nodes configuration file (by default `odnodes.xml`).

- Use multiple `execDeploymentTask` elements, each of which can reference a named `definition` element. That element in turn specifies a grouping of one or more source file locations with a single target file location:

  ```
  <deployment ...>
     <execDefinitionTask useDefinition="MyFirstDef" />
     <execDefinitionTask useDefinition="MySecondDef" />
  </deployment>
  ```

  The `useDefinition` attribute value in an `execDefinitionTask` element references a particular `definition` elements in the configuration, and deploys files specified within that `definition` element when the deployment is run.

EasyDeploy

Fan-out deployments are not supported by the EasyDeploy base server software. To use encryption, you must upgrade to the full-feature base server software.

Transactional Targets in Fan-Out Deployments

Because it is often required that all the target hosts in a fan-out deployment are mirror images of each other as well as the source host, it might be preferable not to deploy files to any of the target hosts in a fan-out deployment, unless a percentage of the targets, or one or more targets in particular, successfully receive the files. If any target does not receive its deployed files successfully, that deployment is considered to be a failure. You can restore those targets that did receive deployed files back to their previous existing states by enabling the transactional feature. See “Transactional Deployments” on page 144 for more information.

You can designate all target hosts to be transactional in a fan-out deployment by specifying a value of `yes` for the `transactional` attribute in the `deployment` element of the fan-out deployment configuration. For example:

```
<deployment transactional="yes">
```

In Figure 33, the source host *mars* performs a deployment-wide transactional fan-out deployment to two target hosts: *jupiter* and *venus*. Although the deployment to *jupiter* was successful, the deployment to *venus* failed. Because this fan-out was transactional on a deployment-wide basis, after the deployment was determined to be a failure, *mars* automatically restores both the target hosts.

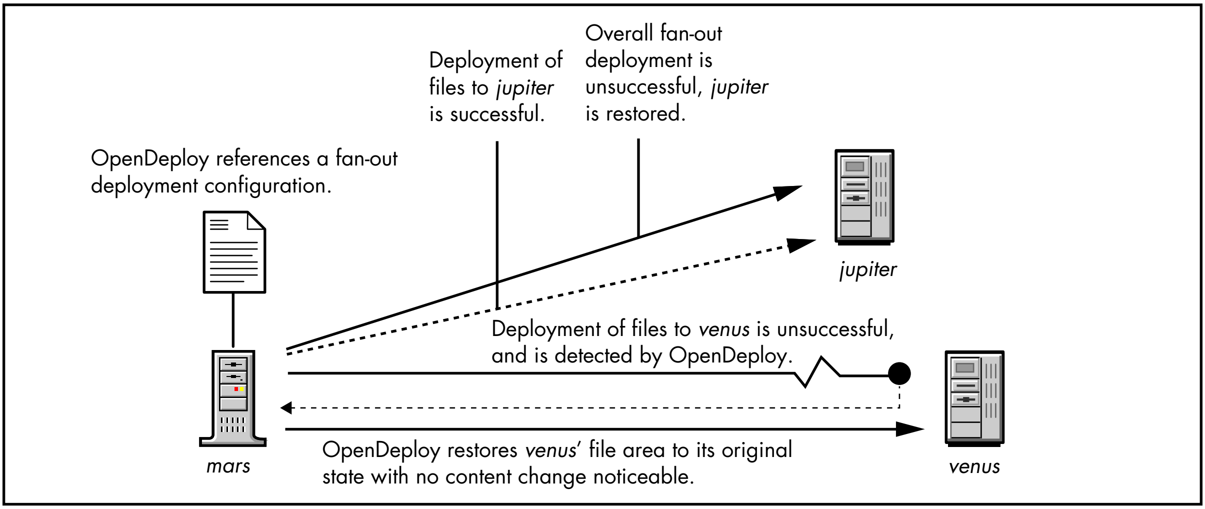

Overall fan-out deployment is unsuccessful, *jupiter* is restored.
Deployment of files to *jupiter* is successful.
OpenDeploy references a fan-out deployment configuration.
jupiter
Deployment of files to *venus* is unsuccessful, and is detected by OpenDeploy.
mars
OpenDeploy restores *venus*' file area to its original state with no content change noticeable.
venus

Figure 33: Transactional Fan-Out Deployment

Determining the Success Criteria

A fan-out deployment can be considered successful even if all the target hosts do not receive the deployed files successfully. You can specify how many targets of a fan-out deployment must receive the deployment successfully for the overall deployment to be considered a success using the quorum feature. A *quorum* is a value specified in the deployment configuration that states a minimum number of targets that must receive all the deployed files successfully for OpenDeploy to consider the deployment a success. A deployment deemed as successful in this manner will not trigger the transactional feature (if enabled) that rolls back the deployment and restores all the fan-out targets to their previous states. If the quorum number is not met, all the target's deployments are rolled back.

You can specify the quorum number as a value for the quorum attribute in the replicationFarm element. For example:

```
<replicationFarm name="fan-out" quorum="2">
   <nodeRef useNode="venus" />
   <nodeRef useNode="jupiter" />
   <nodeRef useNode="mercury" />
</replicationFarm>
```

The number your specify for the quorum can never exceed the number of targets specified as nodeRef element entries in the replicationFarm element. The quorum feature is not supported if the deployment configuration contains multiple replicationFarm elements.

In Figure 34, the deployment has a quorum value of 2, meaning at least two of the targets must receive the deployed files successfully.

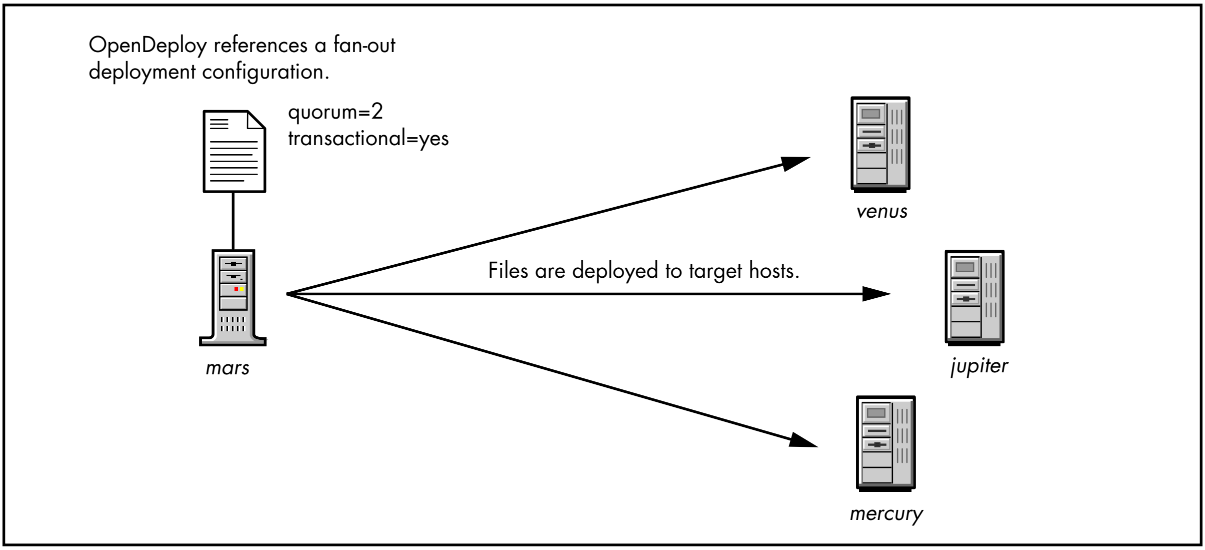

OpenDeploy references a fan-out deployment configuration.
quorum=2
transactional=yes
Files are deployed to target hosts.
venus
jupiter
mars
mercury

Figure 34: Fan-Out Deployment using Quorum and Transactional Features

If the number of successful target deployments does not meet the quorum value, OpenDeploy considers the deployment a failure. Since the transactional feature is enabled, OpenDeploy rolls back the deployment and restores all the hosts, even the one that might have otherwise received its files successfully.

Multi-Tiered Deployments

A *multi-tiered deployment* is a deployment where one or more of the target hosts redeploy the files to a new group of target hosts. Each combination of source host and target hosts is known as a *tier*. Any host that redeploys received files must have the base server software installed to send files. The next-tier source host redeploying the files references its own deployment configuration files for the next-tier deployment.

Each sending host on each tier participating in the multi-tiered deployment must contain the appropriate deployment configuration associated with that tier. The original deployment configuration information is not transmitted from the first source host to the next-tier source host as part of the deployment. Instead, you must configure the deployment at each sending host on each tier prior to running the multi-tiered deployment. Any combination of deployment types and OpenDeploy features can be utilized in a multi-tier deployment. There is also no limit to the number of tiers that can be included, as long as each tier has at least one host with the base server software installed.

In Figure 35, the source host *mars* deploys a fan-out deployment to its target hosts: *venus*, *jupiter*, and *mercury*. This represents the first, or current, tier. The target hosts *venus* and *mercury* have the receiver software installed, allowing them only to receive deployments. However, *jupiter* has the base server installed and can send as well as receive deployed files. After it successfully receives the files deployed from *mars*, *jupiter* references its own deployment configuration file and redeploys the files to its own target hosts: *saturn* and *pluto*. The source host *jupiter* and its targets represent the second, or next, tier.

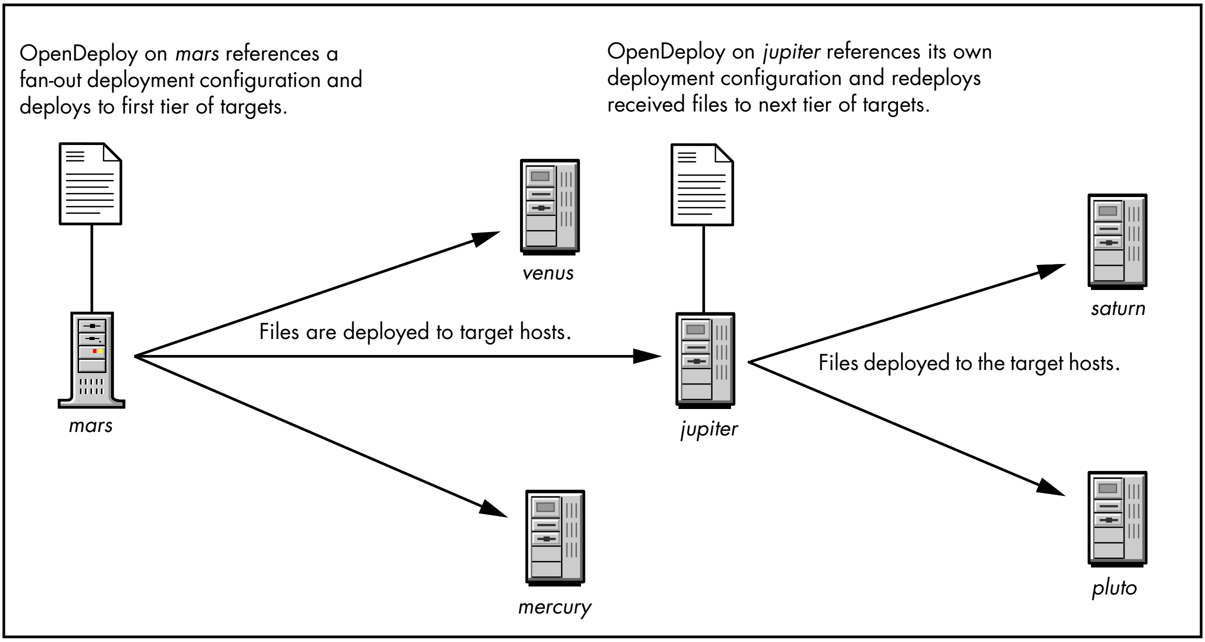

OpenDeploy on *mars* references a fan-out deployment configuration and deploys to first tier of targets.
OpenDeploy on *jupiter* references its own deployment configuration and redeploys received files to next tier of targets.
venus
Files are deployed to target hosts.
mars
jupiter
saturn
Files deployed to the target hosts.
mercury
pluto

Figure 35: Multi-Tiered Deployment

Any target hosts with the base server software installed can start their own deployments, passing on the same deployment data to new targets. This process of redeploying files can continue indefinitely if every tier has a source host capable of redeploying the files it receives.

The role of the source host originating the deployment stops after its target hosts successfully receive the deployed files. The source host at the second tier now takes over, and its deployment configuration file determines the scope and functionality of the deployments on this tier. The second-tier source host also does not receive any deployment configuration information from the current-tier source host.

An OpenDeploy server that is intended to participate in multi-tiered deployments as a secondary- or later-tiered source host must have its deployment configuration file properly configured before the deployment.

A deployment is configured as being multi-tiered by the presence of the nextDeployment element associated with one or more of the sending host's targets as specified in the replication farm:

```
<replicationFarm name="multi-tiered">
   <nodeRef useNode="venus" />
      <nextDeployment
         deployment="tier2_deploy"
         invokeOnSuccess"yes"
         multiTieredTransactional="no"
         />
   <nodeRef useNode="jupiter" />
   <nodeRef useNode="mercury" />
</replicationFarm>
```

A target host of the original deployment that is intended to run a next-tier deployment will have the nextDeployment element and its attributes configured in the target host's nodeRef entry in the replication farm.

The nextDeployment element has the following attributes:

- deployment — enter the name of the deployment that will execute on the target host upon completion of this current deployment. The deployment name will be the same as the deployment configuration file. For example, if the name of the deployment configuration to be invoked is tier2_deploy.xml, then the value would be:

  ```
  deployment="tier2_deploy"
  ```

  That specified deployment configuration must be present on the associated target host for that host to run the next-tier deployment.

- invokeOnSuccess — indicate whether (yes) or not (no) the next-tier deployment should be invoked. If the value is yes, then the next-tier deployment is invoked if the current-tier deployment is successful. If the value is no (the default value), the next-tier deployment is not invoked.

- multiTierTransactional — indicate whether (yes) or not (no) the transactional feature is enabled for the multi-tiered deployment. If enabled, in the event a specified number of targets fail to successfully receive their deployments, the deployments to all the targets are rolled back and their original files are restored. The default value is no. See "Transactional Targets in Multi-Tiered Deployments" on page 153 for more information.

In the following example, a sending host in a multi-tiered deployment has configured its target host *venus* (itself a base server) to run a next-tier deployment after receiving the deployed files:

```
<replicationFarm name="multi-tiered">
   <nodeRef useNode="venus" />
      <nextDeployment
         deployment="monthly"
         invokeOnSuccess"yes"
         />
   ...
</replicationFarm>
```

This configuration indicates the following:

- The deployment configuration that will be used is *monthly*, and that the configuration for monthly (`monthly.xml`) resides in the *`od-home`*`/conf` directory of the node that will redeploy the files.
- The next-tier ending host *venus* only can run the deployment if the overall current-tier deployment is successful.
- The absence of the `multiTieredTransactional` attribute indicates that the deployment is not transactional across tiers. The absence of the attribute is the equivalent of specifying a value of `no`.

EasyDeploy

Multi-tiered deployments are not supported by the EasyDeploy base server software. To use encryption, you must upgrade to the full-feature base server software.

Transactional Targets in Multi-Tiered Deployments

Multi-tiered deployments are supported by the transactional feature. Each sending host on each tier participating in the multi-tiered deployment must have this feature enabled in its respective deployment configuration for the overall deployment to be rolled back in the event the deployment is unsuccessful.

If the criteria for a successful deployment between a single sending host and its targets is not met (as determined by the success criteria), the sending host reports to the sending host at the previous tier from which it received its deployment of the failure. The report of failure is sent back to the originating host where the deployment started, which in turn informs the remaining sending hosts in the subsequent tiers that the deployment failed. This process continues among each sending host across the tiers until all have received the report of failure. OpenDeploy does not commit a multi-tiered transaction until all participating hosts report success. If a report of failure is received, the commit does not take place, and all hosts roll back the deployment and restore the targets to their original states.

The transactional feature is enabled in multi-tiered deployments by the `nextDeployment` element's `multiTierTransactional` attribute.

```
<nextDeployment
   deployment="monthly"
   invokeOnSuccess="yes"
   multiTierTransactional="yes"
   />
```

If you specify a value of `yes` for the `multiTierTransactional` attribute, the next deployment is included in the multi-tier transaction.

The `multiTierTransactional` attribute must be present and enabled in each sending host's deployment configuration for a deployment to participate in the multi-tier transaction.

Figure 36 shows how the transactional feature is used in a multi-tiered deployment. Content is deployed from mars to the targets base server *jupiter* and the receivers *venus* and *mercury*. The base server *jupiter* deploys the content at the next tier to the target receivers *saturn* and *pluto*. Each target indicates success if the files were deployed successfully, or failure if there was a problem. If all deployments are successful, then all targets commit the deployed content. If any sending host reports a failure, the targets roll back the deployed content and restore their files to their previous state.

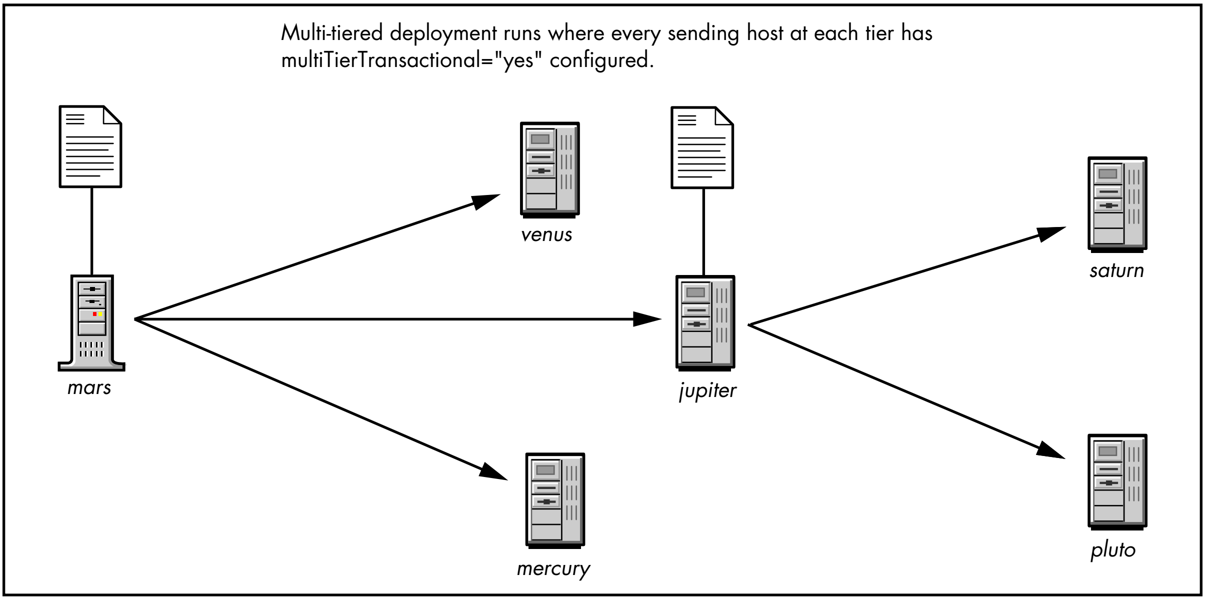

Multi-tiered deployment runs where every sending host at each tier has multiTierTransactional="yes" configured.
venus
mars
jupiter
saturn
mercury
pluto

Figure 36: Use of the Transactional Feature in Multi-Tiered Deployments

Use with Quorum

You can use the quorum feature to specify how many targets must receive their deployments from each sending host at each tier to determine whether the overall deployment is successful or not in a multi-tiered transactional deployment. However, to use the quorum feature, your deployment configuration can contain only one definition. Multiple definitions are not supported. See "Determining the Success Criteria" on page 148 for more information on the quorum feature.

Reverse Deployments

A *reverse deployment* allows new or updated files residing on what is typically a target host (often a production server) to be deployed back to the source host (often a development server) that normally sends deployments. In this relationship, the *reverse host* deploys files to the *reverse target*. Unlike a typical forward deployment, the reverse source can be a host with only the receiver software installed. The reverse target (the host with the base server software installed) manages the reverse deployment, and the reverse source must be listed as a target node in the nodes configuration file of the reverse target host. Reverse deployment files reside in the same directory on the host as other deployments.

Reverse deployments are often used to deploy files generated on production servers back to the development server. For example:

- Web server log files
- Data files created via a CGI application
- Assets uploaded through a Web server application

In Figure 37, the production server *venus* has generated files that need to be reverse deployed back to the development server *mars*. As the reverse target, mars initiates a reverse deployment from the reverse source host venus. The deployment then takes place like any other deployment.

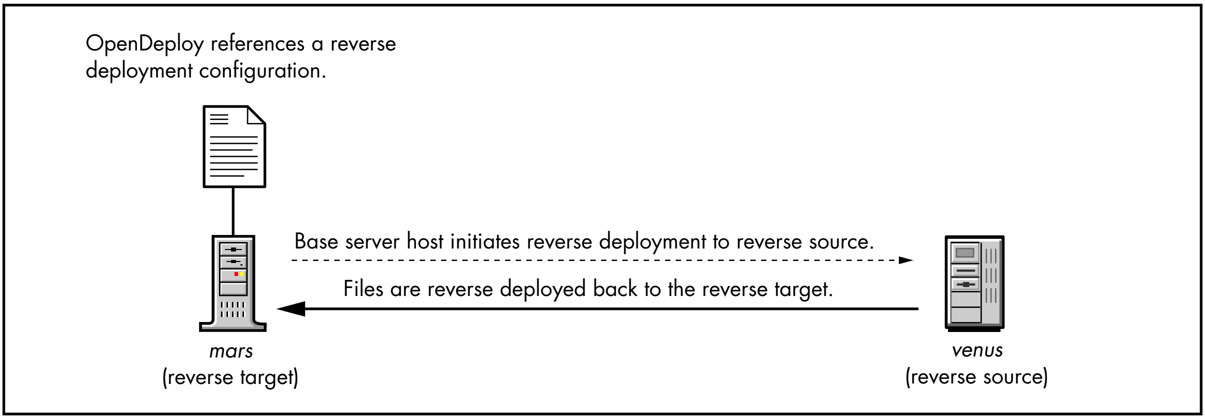

OpenDeploy references a reverse deployment configuration.
Base server host initiates reverse deployment to reverse source.
Files are reverse deployed back to the reverse target.
mars
(reverse target)
venus
(reverse source)

Figure 37: Reverse Deployment

Reverse deployments are configured in a manner similar to traditional deployments, except the source and target host roles are switched as the reverse source and reverse target. The `reverseSource` and `reverseTarget` elements accommodate this change in host roles.

The `reverseSource` element identifies the attributes regarding the sender of the deployment during a reverse deployment. The `useReplicationFarm` attribute specifies the target replication farm to which the reverse source host belongs.

The `reverseTarget` element identifies the attributes regarding the recipient of the deployment during a reverse deployment.

```
<definition name="reverse_deployment">
   <reverseSource useReplicationFarm="MYFARMNAME">
      <sourceFilesystem area="C:\dev\website\files">
         <pathSpecification>
            <path name="monthly" />
         <pathSpecification>
      </sourceFilesystem>
   </reverseSource>
   <reverseTarget>
      <targetFilesystem area="D:\website\files" />
   </reverseTarget>
</definition>
```

Recommended Limits for Number of Deployment Legs

It is recommended that the total number of sending and receiving legs for all active deployments on one system not exceed 50. A "leg" is considered the movement of a specific set of deployed files from a source file location to a target file location on each host (`definition` element). You can compute the number of legs in a deployment by summing the number of node references used in all deployment tasks (`execDeploymentTask` elements) that will be executing simultaneously. Note that in this context a system deploying to itself uses two legs.

In a deployment containing the following target replication farms:

- `myFarm1` (containing four target host node references)
- `myFarm2` (containing five target host node references)

with the following definitions:

- Definition `myDefA` (deploying to `myFarm1`)
- Definition `myDefB` (deploying to `myFarm2`)
- Definition `myDefC` (deploying to `myFarm2`)

and using the following deployment tasks:

- `<execDeploymentTask useDefinition="myDefA" />`
- `<execDeploymentTask useDefinition="myDefB" />`
- `<execDeploymentTask useDefinition="myDefC" />`

the total number of legs would be 14: (4 used by `myDefA`) + (5 used by `myDefB`) + (5 used by `myDefC`)

When creating a deployment configuration, consider whether your deployment involves multiple source-target directory pairings without Deploy and Run triggers per pair. If so, you can employ a multiple sub-source structure to conserve deployment legs. For example:

```
<definition name="oneBigDef">
   <source>
      <sourceFilesystem
         name="S1"
         area="/area1"
         targetArea="/targetarea1"
         >
         <pathSpecification>
            <path name="aaa" />
         </pathSpecification>
      </sourceFilesystem>

      <sourceFilesystem
         name="S2"
         area="/area2"
         targetArea="/targetarea2"
         >
         <pathSpecification>
            <path name="bbb" />
         </pathSpecification>
      </sourceFilesystem>
   </source>
   <target useReplicationFarm="WEBSERVERS">
      <targetFilesystem area="__ignored__" />
   </target>
</definition>
```

Here the number of legs will be equal to the number of node references in the replication farm `WEBSERVERS`.

If the deployment structure requires Deploy and Run per directory pairing, you should use multiple definitions within the deployment configuration. With this approach you gain performance because the deployments execute in parallel, but this comes at the expense of consuming more legs. For example:

```
<definition name="def1">
   <source>
      <sourceFilesystem area="/area1">
         <pathSpecification>
            <path name="aaa" />
         </pathSpecification>
      </sourceFilesystem>
   </source>
   <target useReplicationFarm="WEBSERVERS">
      <targetFilesystem area="/targetarea1" />
   </target>
</definition>

<definition name="def2">
   <source>
      <sourceFilesystem area="/area2">
         <pathSpecification>
            <path name="bbb" />
         </pathSpecification>
      </sourceFilesystem>
   </source>
   <target useReplicationFarm="WEBSERVERS">
      <targetFilesystem area="/targetarea2" />
   </target>
</definition>
```

In this example, the number of legs is twice the number of node references in the replication farm `WEBSERVERS`. This is because there are two definitions that run in parallel.

See “Deploy and Run” on page 201 for more information on this feature.

Using Example and Sample Configurations

OpenDeploy provides a variety of configuration examples and samples. You can use these examples and samples both to learn and better understand the deployment configuration structure and syntax, and also as the basis for composing your own deployments. Because XML-based deployment configuration files must be syntactically correct to work, writing new deployment configuration files can result in a large amount of troubleshooting. By knowing which parts of an example or sample deployment configuration file to modify for your use and which parts to leave alone, you can become productive more quickly and avoid many problems.

Example Configurations

Example configurations are provided for a number of different types of deployments. Each example file contains numerous comments explaining the elements and attributes that make up the configuration.

Example files are located on your OpenDeploy server at the following location:

```
od-home/conf/examples
```

A README file is included describing each example file. Later in this section, you will open one of these example configurations and modify it to use within your enterprise.

Sample Configurations

Sample configurations are a set of deployment configurations designed to allow you to modify them quickly and easily to use within your enterprise. The sample configurations use a common set of variable names that let you modify and test different types of deployments. An accompanying README file provides instructions on how to modify the sample configurations for use within your enterprise.

Sample files are located on your OpenDeploy server at the following location:

```
od-home/conf/examples/samples
```

Customizing an Example Configuration

The example configuration for a simple deployment is `single.xml`, which will deploy files from a source host to a single target host. This is a good sample file from which to learn how to deploy files. Here is the source code for `single.xml`:

```
<deploymentConfiguration>

    <localNode host="MYALLOWSENDINGHOSTNAME"/>

    <replicationFarmSet>
        <replicationFarm name="MYFARMNAME">
            <nodeRef useNode="MyLocalHost"/>
        </replicationFarm>
    </replicationFarmSet>

    <definition name="MYDEFINITIONNAME">
        <source>
            <sourceFilesystem area="/MYOPENDEPLOY/conf/dtd">
                <pathSpecification>
                    <path name="."/>
                </pathSpecification>
            </sourceFilesystem>
        </source>

        <target useReplicationFarm="MYFARMNAME">
            <comparisonRules dateDifferent="yes"/>
            <permissionRules file="0644" directory="0755"/>
            <targetFilesystem area="/MYOPENDEPLOY/tmp"/>
        </target>
    </definition>

    <deployment transactional="no">
        <execDeploymentTask useDefinition="MYDEFINITIONNAME"/>
    </deployment>

    <logRules maxBytes="32Mb" level="verbose"/>

</deploymentConfiguration>
```

You can use this sample file to deploy from a directory on your source host to any other host in your enterprise that has the OpenDeploy base server or receiver software installed. To modify and use this sample configuration to deploy files from your source host, perform the tasks that follow.

List Your Target Hosts in the Nodes Configuration File

Ensure that your source host's node configuration file contains a `node` element entry for your target host. This entry must include the following information:

- The target host's fully qualified DNS host name or IP address.
- The target host's "listener" port. The default value during installation is `20014`.
- A logical name that you designate for the target host. You will use this logical name in the deployment configuration files to indicate the target of the deployment.

Refer to "Defining Target Host Nodes" on page 57 in the *OpenDeploy Installation and Reference Guide* for more information.

Modify the Target Host to Accept Your Deployment

Ensure that your target host's base server or receiver configuration file permits the receiving of deployments from your source host. You must list your source host as a `node` element entry within the `allowedHosts` element. This is similar to the adding of your target host to your node configuration file in the previous task. Refer to "Specifying Allowed Hosts for Received Deployments" on page 74 in the *OpenDeploy Installation and Reference Guide* for more information.

You will also need to specify one or more directory locations on the target host that can receive deployed files from your source host. Refer to "Specifying Allowed Directories for Deployments" on page 76 in the *OpenDeploy Installation and Reference Guide* for more information.

Specify Your Source Host's Name

Add the fully qualified host DNS host name or IP address of your source host in the `localNode` element:

```
<localNode host="MYALLOWSENDINGHOSTNAME" />
```

For example, if your host's IP address was 114.342.23.21, then the `localNode` element would be:

```
<localNode host="114.342.23.21" />
```

Specify the Target Host's Logical Name

Add the `nodeRef` element's `useNode` attribute value to reflect the logical name of your target host.

Each deployment configuration contains a `replicationFarmSet` element and its child elements.

```
<replicationFarmSet>
   <replicationFarm name="MYFARMNAME">
      <nodeRef useNode="MyLocalHost" />
   </replicationFarm>
</replicationFarmSet>
```

You can give the `replicationFarm` element a unique name value for its `name` attribute, but here it is not necessary. The value can remain `MYFARMNAME`.

Each target host is listed within the `replicationFarm` element as a separate `nodeRef` element. Since this is a single target deployment, you only need to include that target's logical name as the value for the `useNode` attribute. If your target host has a logical name of *venus* specified in your server's nodes configuration file, then use that logical name here:

```
<nodeRef useNode="venus" />
```

Specify the Deployment Type

The definition element and its child elements indicate the following information:

- Information about the source of the deployment.
- Information about the targets of the deployment.

You can give the definition element a unique value for its name attribute, but here it is not necessary. The value can remain MYDEFINITIONNAME.

Within the definition element is the source element.

```
<definition name="MYDEFINITIONNAME">
   <source>
      <sourceFilesystem area="/MYOPENDEPLOY/conf/dtd">
            <pathSpecification>
               <path name = "." />
            </pathSpecification>
      </sourceFilesystem>
   </source>
   <target useReplicationFarm="MYFARMNAME">
      <comparisonRules dateDifferent="yes" />
         <permissionRules file="0644" directory="0755" />
      <targetFilesystem area="/tmp" />
   </target>
</definition>
```

The source element contains all the elements and attributes that indicate the following:

- Type of deployment that is occurring.
- Directory or TeamSite area where the source files are located.
- Special features such as modifications to the deployment criteria.

Our example is of a directory comparison deployment. A directory comparison deployment will compare the files residing in a specified location on the source host with the files residing in a specified location on the target host. Those files that meet the deployment criteria based on differences in size, modification date, ownership (UNIX only), and other conditions, are moved to the specified target host location. A directory comparison deployment is indicated by the presence of the `sourceFilesystem` and `targetFilesystem` element in the configuration. See "Directory Comparison Deployments" on page 124 for more information.

Specifying the Source Host File Locations

Within the `sourceFilesystem` element is the `area` attribute. Here is where the location of files on the source host are specified.

```
<source>
   <sourceFilesystem area="/MYOPENDEPLOY/conf/dtd">
      <pathSpecification>
         <path name = "." />
      </pathSpecification>
   </sourceFilesystem>
</source>
```

The `area` attribute's value is the full path to the directory containing the files that will participate in the deployment. For example:

`area="C:\reports"`*or*

`area="/etc/reports"`

You can specify a subdirectory within the source file location using the `path` element and its `name` attribute. This value is a subdirectory or a path to a subdirectory relative to the area location. However, in this example no further location within the source area is required, so the `path` element's `name` attribute value is "`.`"

Specify the Target Host File Location

You must specify the location on the target host where the deployed files will reside following the deployment.

```
<target useReplicationFarm="MYFARMNAME">
   <comparisonRules dateDifferent="yes" />
   <permissionRules file="0644" directory="0755">
   </permissionRules>
   <targetFilesystem area="/tmp" />
</target>
```

The `target` element and its child elements and attributes include the location where the deployed files will reside on the target host after the deployment, and any optional features you want to augment the deployment criteria. The `target` element's `useReplicationFarm` attribute value must match the `replicationFarm` element's `name` attribute value you specified earlier. In this example you can leave it as `MYFARMNAME`.

In this example, the `permissionRules` element specifies the permissions to be set on deployed files and directories on a UNIX system. However, that is a more advanced topic, so it will not be covered here.

The target location is specified in the `targetFilesystem` element's `area` attribute. In other types of deployments, you can specify the target location as a TeamSite area instead. The `targetFilesystem` element's `area` attribute value is the full file system path to where the deployed files will reside on the target host. Enter the path to the directory where you want the deployed files to reside, for example:

`area="C:\reports\western"`*or*

`area="/etc/reports/western"`

Specify Transactional

In the final section of our example deployment is the `deployment` element and its child elements and attributes.

```
<deployment "transactional="no">
   <execDeploymentTask useDefinition="MYDEFINITIONNAME" />
</deployment>
```

The `transactional` feature will automatically cause OpenDeploy to roll back a deployment and restore the target host files to their previous state in the event one or more target deployments are unsuccessful. However, for this example deployment, the feature is not needed. You can leave the default value of `no` as is.

Modifying the Sample Configuration to Fit Your Needs

Now that you have covered the important parts of the sample deployment configuration, you can modify its use in your own enterprise. In the next example, the original deployment configuration has been modified to reflect the following setup of an actual OpenDeploy source host using the following values. The element and attribute containing the changed value are noted in parentheses:

- Host name of the OpenDeploy server sending the deployment (`localNode host`):

  ```
  114.342.23.21
  ```

- Logical name of target host (`nodeRef useNode`):

  ```
  venus
  ```

- Location on source host where files reside (`sourceFilesystem area`)

  ```
  C:\reports
  ```

- Location on target host where deployed files will reside (`targetFilesystem area`)

  ```
  C:\reports\western
  ```

The following example shows the code with the changed values.

```
<deploymentConfiguration>

  <localNode host="114.342.23.21" />

  <replicationFarmSet>
      <replicationFarm name="MYFARMNAME">
          <nodeRef useNode="venus" />
      </replicationFarm>
  </replicationFarmSet>

  <definition name="MYDEFINITIONNAME">
      <source>
          <sourceFilesystem area = "C:\reports">
              <pathSpecification>
                  <path name = "." />
              </pathSpecification>
          </sourceFilesystem>
      </source>
      <target useReplicationFarm="MYFARMNAME">
          <comparisonRules  dateDifferent="yes" />
          <permissionRules  file="0644" directory="0755">
          </permissionRules>
          <targetFilesystem area="C:\reports\western" />
      </target>

  </definition>

  <deployment transactional="no">
      <execDeploymentTask useDefinition="MYDEFINITIONNAME" />
  </deployment>

</deploymentConfiguration>
```

To use this example, you must also modify the base server or receiver configuration file of the host receiving the deployment (depending on whether that recipient host has the base server or receiver software installed).

After you update the recipients host's server configuration, you must reset it either by restarting OpenDeploy, or by using the `iwodserverreset` command-line tool. See "Starting OpenDeploy" on page 47 and "Refreshing the OpenDeploy Server" on page 57 for more information.

After you have finished modifying the sample deployment configuration file to function within your OpenDeploy environment, place some files in the source area directory and try running a deployment. See “Starting a Deployment” on page 87 for more information.

After you have become comfortable with how to modify a sample file to work within your OpenDeploy environment, you can make more modifications to the `single.xml` configuration file used in the example, or try some of the other example files. You can add additional target hosts to the nodes configuration file and the `nodeRef` element and turn your single deployment into a multi-target fan-out deployment. You can also modify the `source` and `target` elements, and change the deployment type from a directory comparison to a TeamSite comparison or file list deployment. Finally, you can also add additional features such as filters or rules described later in this manual.

Chapter 4

Deployment Features

This chapter describes the following deployment features and how they are configured:

- Filters
- Comparison rules
- Transfer rules
- Permission rules
- Use with access control lists (ACLs)
- Deploying symbolic links
- Parameter substitution
- Deploy and Run scripting
- Use with adapters

Filters

You can modify the deployment configuration to include and exclude files and directories from the deployment through the use of filters. Filters refine the deployed content to only those items you want included. They can be used at the source of the deployment, one or more targets, or a combination of the two.

These filters can use one or both of the following criteria to determine whether to deploy an item or not:

- File system path location and name
- Naming pattern using regular expression

Filters are applied differently depending on the deployment type. The following table describes how filters work on each type of deployment.

Deployment Type	Filtering Action
Directory comparison	Filters can be configured to be applied to either only the source-side, or to both the source-side and target-side. Source-side filters are applied to the source content resulting in a refined source image. Target-side filters are applied to the target content resulting in a refined target image. These two images are compared, and the result of the compare is deployed. The common configurations for directory comparison filters are: 1. Identical filters are specified for both source-side and target-side, so that the source and target contents are identical. This configuration allows an exact copy of all the content that passes the filters on both sides, but it leaves everything else alone on the target. 2. Only source-side filters are used, so that the target is a refined version of the source content. This configuration provides no protections to the target. If the DoDeletes feature is enabled, the target will become an exact mirror of the refined source content.
TeamSite comparison	Filters are applied to the resulting list of paths produced from the TeamSite comparison of the area and previousArea file locations. The resulting filtered list is deployed.
File list	Filters are applied to the list of relative paths in the file list. The resulting filtered list is deployed.

Filter Placement

Filters are configured within the `filters` element throughout a deployment configuration. You have several options as to the placement of filters. The following sections describe the placement of filters within the configuration.

Source-Side Filters

Source-side filters can appear both within the `pathSpecification` and the `targetRules` elements. For example:

```
<source>
   <sourceFilesystem area="C:\dev\website\files">
      <pathSpecification>
         <path name="." />
         <filters>
            ...
         </filters>
         <targetRules>
            <filters>
               ...
            </filters>
         </targetRules>
      </pathSpecification>
   </sourceFilesystem>
</source>
```

Target-Side Filters

Target-side filters can appear both within the `target` element:

```
<target>
   <targetFilesystem area="D:\website\files" />
   <filters>
      ...
   </filters>
</target>
```

and within a `targetRules` element at the node level:

```
<replicationFarmSet>
   <replicationFarm name="MYREPLICATIONFARM">
      <nodeRef useNode="venus">
         <targetRules>
            <filters>
               ...
            </filters>
         </targetRules>
      </nodeRef>
   </replicationFarm>
</replicationFarmSet>
```

Override Precedence

Filters specified for the `targetRules` element within the `pathSpecification` element override any filters specified for the `target` element.

Filters specified for the `targetRules` element within the `nodeRef` element override any filters specified for the `targetRules` element within the `pathSpecification` element and any filters specified for the `target` element.

Inclusion Filters

Inclusion filters allow you to configure one or more criteria based on file system paths or naming patterns to filter only those files and directories you want included in the deployment. You can use multiple occurrences of either file system- or pattern-based inclusion filters, but you cannot combine both types in the same deployment configuration. You also cannot combine either type of inclusion filter with any type of exclusion or exception filter. See “Combining Filter Types” on page 181 for more information on filter compatibility.

If no inclusion filters are present, all files are included in the deployment, except any that are subsequently blocked from the deployment by exclusion filters. The use of exclusion filters is described later in this section.

File System-Based Inclusion Filters

File system-based inclusion filters permit those directories and files that match the specified path criteria to be deployed. Paths specified are relative to the source file location of the deployment, such as a file system directory or TeamSite edition.

File system-based inclusion filters are configured in the `includePath` element within the `filters` element:

```
<filters>
   <includePath subPath="path_to_be_included" />
</filters>
```

The `includePath` element's `subPath` attribute specifies those paths and file names that represent the inclusion criteria. Those items to be deployed must meet that criteria in order to be included in the deployment. The path specified in the `subPath` attribute is relative to the local directory or TeamSite area on the host containing the files to be filtered, as specified in the `area` attribute value of the `sourceFilesystem` or `sourceTeamsite` elements, respectively.

The following example shows a deployment that would only include the contents of the directory `reports/monthly`:

```
<filters>
   <includePath subPath="reports/monthly" />
</filters>
```

You can add inclusion paths to your deployment configuration file by adding another `includePath` element for each included path. For example:

```
<filters>
   <includePath subPath="reports/monthly" />
   <includePath subPath="reports/quarterly" />
</filters>
```

When file system-based inclusion filters are present in the deployment configuration, an item needs only to match a single filter to be included.

File system-based inclusion filters are incompatible with pattern-based inclusion filters, as well as all exclusion and exception filters, in the same deployment configuration.

Pattern-Based Inclusion Filters

Pattern-based inclusion filters permit those directories and files that match the specified naming pattern to be deployed.

Inclusion filters are configured in the `includePattern` element within the `filters` element:

```
<filters>
   <includePattern regex="pattern_to_be_included" />
</filters>
```

The `includePattern` element's `regex` attribute specifies the regular expression naming pattern against which the files and directories in a deployment are compared. Those items that match the naming pattern are included in the deployment. Those that do not are not included. See "Supported Regular Expressions" on page 182 for guidelines on OpenDeploy use and support of regular expressions.

The following example shows a deployment that would only include files with the extension `.html`:

```
<filters>
   <includePattern regex=".*\.html$" />
</filters>
```

You can specify multiple inclusion patterns for a deployment configuration file by adding another `includePattern` element for each pattern. For example:

```
<filters>
   <includePattern regex=".*\.html$" />
   <includePattern regex=".*\.txt$" />
</filters>
```

When composing your regular expressions, follow your operating system's path separator syntax. For example, UNIX uses "`/`" while Windows uses "`\\`". You can specify matches for a file on both Windows and UNIX operating systems by including `[/\\]` in your regular expression. This allows you to overcome the path separator syntax differences between Windows and UNIX. For example, to provide for a match of the file `index.html` on both Windows and UNIX file systems, you would configure it in the following way:

```
regex="^\.[/\\]index\.html$"
```

When pattern-based inclusion filters are present in the deployment configuration, an item needs only to match a single filter to be included.

Pattern-based inclusion filters are incompatible with file system-based inclusion filters, as well as all exclusion and exception filters, in the same deployment configuration.

Exclusion Filters

Exclusion filters allow you to configure one or more criteria based on paths or naming patterns to filter out those files and directories you do not want included in the deployment. You can configure multiple exclusion filters of both file system- and pattern-based types in the same configuration. If only exclusion filters are in the configuration, items that meet the exclusion criteria are not deployed. If multiple exclusion filters are present, an item needs only to match a single one to be excluded. Exclusion filters are incompatible with inclusion filters. See "Combining Filter Types" on page 181 for more information on filter compatibility.

File System-Based Exclusion Filters

File system-based exclusion filters prevent those directories and files that match the specified path and name criteria from being deployed.

File system-based exclusion filters are specified by the `excludePath` element within the `filters` element:

```
<filters>
   <excludePath subPath="path_to_be_excluded" />
</filters>
```

The `excludePath` element's `subPath` attribute specifies the file system location and name criteria against which the files and directories in a deployment are compared. Those items that match are excluded from the deployment. The path specified in the `subPath` attribute is relative to the local directory or TeamSite area on the host containing the files to be filtered, as specified in the `area` attribute value of the `sourceFilesystem` or `sourceTeamsite` elements, respectively.

The following example shows a deployment that would exclude the directory `WebFiles/working`:

```
<excludePath subPath="WebFiles/working" />
```

You can specify multiple exclusion paths for a deployment by adding another `excludePath` element for each excluded path. For example:

```
<filters>
   <excludePath subPath="WebFiles/working" />
   <excludePath subPath="WebFiles/intranet" />
</filters>
```

Pattern-Based Exclusion Filters

Pattern-based exclusions filters prevent those directories and files that match the specified naming criteria from being deployed. Those items that do not match the exclusion criteria are deployed.

Pattern-based exclusion filters are specified by the `excludePath` element within the `filters` element:

```
<filters>
   <excludePattern regex="pattern_to_be_excluded" />
</filters>
```

The `excludePattern` element's `regex` attribute specifies the regular expression naming pattern against which the files and directories in a deployment are compared. Those items that match are excluded from the deployment. See "Supported Regular Expressions" on page 182 for guidelines on OpenDeploy use and support of regular expressions.

For example, if you wanted to exclude any file whose name includes the extension `.html`, the `excludePattern` element value would be:

```
<excludePattern regex=".*\.html$" />
```

You can specify multiple exclusion patterns for a deployment by adding another `excludePattern` element for each excluded pattern. For example:

```
<filters>
   <excludePattern regex=".*\.html$" />
   <excludePattern regex=".*\.txt$" />
</filters>
```

Exception Filters

In some cases, you might want to protect certain items or classes of items that would otherwise be excluded from the deployment by the presence of exclusion filters. For example, if you wanted to exclude any file whose name contains `internal` from the deployment, but still deploy the file `internal.html`, you would need to configure an exclusion filter for all files named `internal`, but also contain an exception for the file `internal.html`. File system- and pattern-based exclusion filters provide overrides to any and all exclusion filters present in the configuration.

Exception filters only work with the presence of exclusion filters in the deployment. You can use multiple occurrences of either file system- or pattern-based exception filters, but you cannot combine both types in the same deployment configuration. Exception filters are incompatible with any type of inclusion filter. See “Combining Filter Types” on page 181 for more information on filter compatibility.

File System-Based Exception Filters

File system location-based exception filters protect those directories and files that match the specified path and name criteria from being excluded from the deployment.

File system-based exception filters are specified by the `exceptPath` element within the `filters` element:

```
<filters>
   <excludePath subPath="path_to_be_excluded" />
   <excludePattern regex="pattern_to_be_excluded" />
   <exceptPath subPath="path_to_be_excepted" />
</filters>
```

The `exceptPath` element’s `subPath` attribute specifies the file system location and name criteria against which the files and directories in a deployment are compared. Those items that match are protected from exclusion filters. The path specified in the `subPath` attribute is relative to the local directory or TeamSite area on the host containing the files to be filtered, as specified in the `area` attribute value of the `sourceFilesystem` or `sourceTeamsite` elements, respectively. Follow

In the following example:

```
<filters>
   <excludePattern regex=".*\.html$" />
   <exceptPath subPath="reports/monthly/index.html" />
</filters>
```

Those files within the directory `reports/monthly`:

```
reports/monthly/reports.html
reports/monthly/comments.html
```

are excluded from the deployment, but the file:

```
reports/monthly/index.html
```

is retained in the deployment because the exception filter overrides the exclusion filter.

File system-based exception filters can be used in any number, and with any combination of file system- and pattern-based exclusion filters. However, file system-based exception filters are incompatible with pattern-based exception filters.

Pattern-Based Exception Filters

Pattern-based exclusions filters prevent those directories and files that match the specified naming criteria from being excluded from the deployment.

Pattern-based exception filters are specified by the `exceptPattern` element within the `filters` element:

```
<filters>
   <excludePath subPath="path_to_be_excluded" />
   <excludePattern regex="pattern_to_be_excluded" />
   <exceptPattern regex="pattern_to_be_excepted" />
</filters>
```

The exceptPattern element's regex attribute specifies the regular expression naming pattern against which the files and directories in a deployment are compared. Those items that match are protected from exclusion filters. See "Supported Regular Expressions" on page 182 for guidelines on OpenDeploy use and support of regular expressions.

For example, if you wanted to protect any file whose name contains the extension .html from any exclusion filters in the deployment, the exceptPattern element value would be:

```
<exceptPattern regex=".*\.html$" />
```

In the following example:

```
<filters>
   <excludePath subPath="reports/monthly" />
   <exceptPattern regex="regex=".*\.html$" />
</filters>
```

Those files within the directory reports/monthly:

```
reports/monthly/reports.doc
reports/monthly/reports.pdf
reports/monthly/reports.txt
```

are excluded from the deployment, but the file:

```
reports/monthly/reports.html
```

is retained in the deployment because the exception filter overrides the exclusion filter.

Combining Filter Types

OpenDeploy employs the following rules for combining the different types of filters in a deployment configuration.

Inclusion Filters

File system-based and pattern-based inclusion filters are incompatible with each other, and with all other types of filters. A deployment can contain multiple occurrences of either type of inclusion filter, but no others.

Exclusion Filters

You can combine both file system- and pattern-based exclusion filters in any combination and number within the same deployment configuration.

Exclusion filters are incompatible with any type of inclusion filter.

You can combine exclusion filters with either file system- or pattern-based exception filters, but not both.

Exception Filters

You can use either file system or pattern-based exception filters, but not both, in a deployment configuration containing exclusion filters. File system and pattern-based exception filters are mutually exclusive to each other.

Exception filters are incompatible with any type of inclusion filter.

Specifying Path Syntax for Filters

When specifying a path for either file system- or pattern-based filters, use the path delimiter syntax for the host's operating system ("\" for Windows, "/" for UNIX). If you are deploying to a mix of Windows and UNIX target hosts, the UNIX path delimiter syntax ("/") will work for both Windows and UNIX targets.

Supported Regular Expressions

OpenDeploy supports POSIX 1003.2 extended regular expressions (ERE), and makes use of Henry Spencer's NFA regex package. If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

When composing your regular expressions, follow your operating system's path separator syntax. For example, UNIX uses "/" while Windows uses "\\". You can specify matches for a file on both Windows and UNIX operating systems by including `[/\\]` in your regular expression. This allows you to overcome the path separator syntax differences between Windows and UNIX.

For example, to provide for a match of the file `index.html` on both Windows and UNIX file systems, you would configure it in the following way:

```
regex="^\.[/\\]index\.html$"
```

The following table summarizes supported regular expression characters and describes their functions:

Character	**Function**
`\`	Indicates next character should not be interpreted literally if it normally is, and should be interpreted literally if it normally is not.
`^`	Matches beginning of line.
`$`	Matches end of input or line.
`*`	Matches 0 or more instances of preceding character.
`+`	Matches 1 or more instances of preceding character.
`?`	Matches 0 or 1 instances of preceding character.
`.`	Matches any single character.
`x|y`	Matches either `x` or `y`.
`{n}`	Matches exactly `n` instances of preceding character (where n is an integer).
`{n,}`	Matches at least `n` instances of preceding character (where n is an integer).
`{n,m}`	Matches at least n and at most m instances of preceding character (where `n` and `m` are integers).
`[xyz]`	Matches any one of enclosed characters (specify range using hyphen, such as `[0-9]`.
`[^xyz]`	Matches any character not enclosed (specify range using hyphen, such as `[^0-9]`.
`\n`	Matches a line feed.
`\t`	Matches a tab.

Target Host File Deletions Using Filtered Deployments

A file or directory that is excluded from a filtered deployment is treated as being non-existent at the source host. If the `doDeletes` attribute of the `transferRules` element has a value of `yes`, then the same file at the target host will be deleted during the deployment. This is because the `doDeletes` attribute removes any target host file that does not have an equivalent file at the source host. Filtered deployments can inadvertently cause target-side files to be deleted in this manner.

Filters specified on the target side cause those files to be ignored during comparisons. This results in files with the same name on the source side being deployed since they appear to be missing on the target side. Be careful if you are using the `doDeletes` feature in conjunction with filtered deployments. See “File Transfer Rules” on page 186 for more information.

There are special rules for using the `doDeletes` option with file list deployments. See “Using doDeletes with File List Deployments” on page 139 for more information.

File Comparison Rules

Modification date is the primary comparison criterion used to determine whether or not a given file should be deployed.

If a source-side file’s modification date is newer than its target-side equivalent, then the file is deployed. You can also configure the deployment to deploy source-side files that have modification dates older than its target-side equivalent using the `revert` option, or if there is a difference in modification date either way using the `dateDifferent` option. These options are described later in this section.

If a source-side file’s modification date is identical to its target-side equivalent, then the following criteria are used in sequential order to determine whether or not a given file should be deployed:

- Type mismatch (a file and a directory sharing the same name)
- User difference (UNIX only)
- Group difference (UNIX only)
- Permission difference (UNIX only)

- Access control list (ACL) difference (Windows only, disabled by default)
- Size difference

If either the source-side's or target-side's, or both, host's operating systems do not support a particular comparison criterion, that criterion is skipped.

You can customize some these criteria within a deployment configuration by setting various attributes within the `comparisonRules` element. Here is a listing of those attributes:

- `dateDifferent` — indicate whether (`yes`) or not (`no`) a file should be deployed if there is any difference in file date (older or newer) between the source and target versions. This differs from the OpenDeploy default date-based comparison setting, where a file is deployed only if the source file is newer than the target file. A value of `yes` indicates that the file should be deployed. Default value is `no`. The `dateDifferent` attribute is mutually exclusive with the `revert` attribute.
- `revert` — indicate whether (`yes`) or not (`no`) a file should be deployed if the source version is older than the target version. A value of `yes` indicates that the file should be deployed. Default value is `no`. The `revert` attribute is mutually exclusive with the `dateDifferent` attribute.
- `ignoreAcls` (Windows only) — indicate whether (`yes`) or not (`no`) to ignore differences in the ACLs during the file comparison. Default value is `yes`. See "Using OpenDeploy with ACLs" on page 194 for more information.
- `ignoreModes` (UNIX only) — indicate whether (`yes`) or not (`no`) to ignore differences in the UNIX-based permission bit mask during the file comparison. Default value is `no`.
- `ignoreUser` (UNIX only) — indicate whether (`yes`) or not (`no`) to ignore differences in the UNIX-based file user ownership during the file comparison. Default value is `no`.
- `ignoreGroup` (UNIX only) — indicate whether (`yes`) or not (`no`) to ignore differences in the UNIX-based file group ownership during the file comparison. Default value is `no`.

You can enable and disable comparison rules in any combination. For example, in the following occurrence of the `comparisonRules` element:

```
<comparisonRules dateDifferent="yes" ignoreAcls="yes" />
```

OpenDeploy applies the following rules:

- A file will be considered for deployment if the source file modification date is either older or newer than the target file.
- Differences in access control list (ACL) settings between the source are ignored during the comparison.

Otherwise, all other comparison criteria are in effect.

Access options specific to UNIX are ignored when deploying to a Windows target host and access options specific to Windows are ignored when deploying to a UNIX target host. Therefore, you will not receive any errors if you define both:

`ignoreAcls="yes"` *and*

`ignoreUser="yes"`

File Transfer Rules

You can modify your deployment configuration to follow or ignore various file transfer-related rules through the `transferRules` element and its various attributes. Here is a listing of those attributes:

- `doDeletes` — indicate whether (`yes`) or not (`no`) files and directories that reside in the target host area but not in the source host area should be deleted following the deployment. By default they are not. Default value is `no`.

  There are special rules for using the `doDeletes` option with file list deployments. See “Using doDeletes with File List Deployments” on page 139 for more information.

  This feature sometimes can cause inadvertent file deletions when used in conjunction with target-side filters. See “Target Host File Deletions Using Filtered Deployments” on page 184 for more information.

- `dontDo` — indicate whether (`yes`) or not (`no`) to proceed with the deployment following the comparison. If the value is `yes`, the deployment will not occur. Default value is `no`.

  Performing a deployment using this feature will log all simulated deployed files to the deployment log. This is a good tool to use to check and compare files without actually performing a deployment. Files that are logged as being deployed indicate a difference between what is on the source server and the target server. You can also enable this feature using the `iwodstart -sim` command-line tool, or the simulated deployment feature in the OpenDeploy user interface. See "Performing a Simulated Deployment" on page 91 for more information on the benefits of simulated deployments.

- `preserveAcls` (Windows only) — indicate whether (`yes`) or not (`no`) to preserve the Windows access control lists (ACLs) when the files are moved. By default, OpenDeploy applies ACLs based on the ACLs already existing on the containing folders on the target host receiving the deployed files. Default value is `no`. See "Using OpenDeploy with ACLs" on page 194 for more information.

  The `preserveAcls` and `setAccess` attributes are mutually exclusive. Do not enable both in the same configuration. If both are enabled, the `setAccess` attribute is honored and `preserveAcls` attribute is ignored. See "File Permission Rules" on page 190 for more information about the `setAccess` attribute.

  The `preserveAcls` and `changeAccess` attributes are also mutually exclusive. Do not enable both in the same configuration. If both are enabled, the `changeAccess` attribute is honored and `preserveAcls` attribute is ignored. See "File Permission Rules" on page 190 for more information about the `changeAccess` attribute.

- `followLinks` (UNIX only) — indicate whether (`yes`) or not (`no`) symbolic links on the source and target hosts will be followed when the files are moved. Default value is `no`.

  Enabling the `followLinks` attribute within the `transferRules` element only affects the target side. If you want to apply this feature to the source side as well, you must enable it in the `sourceTransferRules` element. See "Deploying Symbolic Links" on page 197 for more information.

- `svrTryCount` (Windows only) — enter the number of times OpenDeploy will attempt to deploy the file to the target host. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.

- `svrTryInterval` (Windows only) — enter the amount of time in seconds OpenDeploy waits between deployment attempts. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.
- `svrTryDisableOverwrite` (Windows only) — indicate whether (`yes`) or not (`no`) to disable the ability of OpenDeploy to deploy files to a server even if the `svrTryCount and svrTryInterval` elements are specified. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic. Default value is `no`.
- `rmReadOnly` (Windows only) — indicate whether (`yes`) or not (`no`) you want a deployed file to be able to overwrite its read-only target equivalent. If this feature is enabled with a value of `yes`, OpenDeploy will remove the read-only attribute from the target file, allowing the deployment to occur. A value of no will prevent the overwriting. Default value is `no`.
- `compression` — indicate whether (`yes`) or not (`no`) content is compressed prior to being deployed.
- `compressionLevel` — indicate what level (`0-9`) of compression OpenDeploy uses if compression of deployed content is enabled. A value of `1` is the lowest level of compression, and `9` is the highest. A value of `0` provides no compression at all.

You can enable and disable comparison rules in any combination. For example, in the following occurrence of the `transferRules` element:

```
<transferRules doDeletes="yes" preserveAcls="yes" />
```

OpenDeploy applies the following rules:

- Any files existing in the target host area but not in the corresponding source area will be deleted following the deployment.
- (Windows only) Access control list (ACL) information will be retained following the deployment.
- Access options specific to UNIX are ignored when deploying to a Windows target host and access options specific to Windows are ignored when deploying to a UNIX target host.

Compression

You have the option of compressing content being deployed. Compression can reduce the size of the deployment, saving network bandwidth and deployment time between the source and targets. However, compression and associated decompression can require more CPU time or power.

Compression is configured in the `compression` and `compressionLevel` attributes of the `transferRules` element:

```
<transferRules
    ...
    compression="yes"
    compressionLevel="6"
    />
```

To enable compression in deployments, specify a value of `yes` for the compression element. The default value is `no`.

If you elect to use compression, you can select the level of compression, with `1` being the lowest and `9` the highest. A value of zero provides no compression at all, even if it is enabled in the compression element.

Whether to use compression or not, and to what level, depends on the system priorities within your enterprise. If bandwidth limitation is an issue, compression can be a valuable asset. If CPU power conservation is more important, compression may not be practical. The compression level should represent the best balance of these factors. A compression level of `6` is recommended for a typical enterprise. This is also the default level used if none is specified in the configuration.

File Permission Rules

You can modify your deployment configuration to follow or ignore various file permission-related rules through the `permissionRules` element and its various attributes. Here is a listing of those attributes:

- `amask` (UNIX only) — enter the bit mask (in octal) to be ANDed with the permission bits of all files and directories. The `amask` octal value combines with the existing permission bit value of the affected file. If a file has the existing permission value of `664` (-rw-rw-r--) and the `amask` attribute as the following value:

  ```
  amask="770"
  ```

  then the resulting permission for that file (664 AND 770) following the deployment would be `660` (-rw-rw----).

- `omask` (UNIX only) — enter the bit mask (in octal) to be ORed with the permission bits of all files and directories. The `omask` octal value combines with the existing permission bit value of the affected file. If a file has the existing permission value of `666` (-rw-rw-rw-) and the `omask` attribute as the following value:

  ```
  omask="022"
  ```

  then the resulting permission for that file (666 OR 022) following the deployment would be `644` (-rw-r--r--).

- `directory` (UNIX only) — enter the permissions (in octal) given to all deployed directories. For example, if you wanted deployed directories to have the permission “drwxrwx---”, then the resulting value would be:

  ```
  directory="770"
  ```

- `file` (UNIX only) — enter the permissions (in octal) given to all deployed files. For example, if you wanted deployed files to have the permission “-rw-rw-r-x” , then the resulting value would be:

  ```
  file="665"
  ```

- `group` (UNIX only) — enter the group assigned to all deployed files and directories. This attribute value must be a valid group name or group ID. For example:

  `group="tech_pubs"` *or*

  `group="200"`

  You must also specify the `user` attribute if you use the `group` attribute.

- `user` (UNIX only) — enter the user who will own all deployed files and directories. This attribute value must be a valid user name or user ID. For example:

  `user="jdoe"` *or*

  `user="105"`

  You must also specify the `group` attribute if you use the `user` attribute.

- `changeAccess` (Windows only) — enter a value that modifies the access control lists (ACLs) so that specified users have the designated rights. The new access control entry (ACE) for each specified user allows only the specified rights, discarding any existing ACE. In the following example:

  ```
  changeAccess="{ jdoe:W, tech_pubs:NONE }"
  ```

  any existing ACEs for `jdoe` and `tech_pubs` are removed, `jdoe` is granted write access, and the group `tech_pubs` has no access at all. Any other access rights that may have existed for other users are left unchanged. See “Using OpenDeploy with ACLs” on page 194 for more information.

  The `changeAccess` and `setAccess` attributes are mutually exclusive. Do not enable both in the same configuration, or an error will occur.

  The `changeAccess` and `preserveAcls` attributes are mutually exclusive. Do not enable both in the same configuration. If both are enabled, the `changeAccess` attribute is honored and `preserveAcls` attribute is ignored. See “File Transfer Rules” on page 186 for more information about the `preserveAcls` attribute.

- `setAccess` (Windows only) — enter a value that replaces the ACLs for the deployed files and directories. In the following example:

  ```
  setAccess="{ jdoe:ALL, tech_pubs:RX }"
  ```

  the existing ACL is removed and the user `jdoe` is granted full access. The group `tech_pubs` has read access to the specified files. Any other access rights that may have existed for the file are removed. See "Using OpenDeploy with ACLs" on page 194 for more information.

  The `setAccess` and `changeAccess` attributes are mutually exclusive. Do not enable both in the same configuration, or an error will occur.

  The `setAccess` and `preserveAcls` attributes are mutually exclusive. Do not enable both in the same configuration. If both are enabled, the `setAccess` attribute is honored and `preserveAcls` attribute is ignored. See "File Transfer Rules" on page 186 for more information about the `preserveAcls` attribute.

Access options specific to UNIX are ignored when deploying to a Windows target host and access options specific to Windows are ignored when deploying to a UNIX target host.

In the following example:

```
<permissionRules
   directory="770"
   file="664"
   group="marketing"
   user="rroe"
   />
```

OpenDeploy applies the following rules:

- The deployed directories will have "drwxrwx---" access as indicated by the value `770`.
- The deployed files will have "-rw-rw-r--" access as indicated by the value `664`.
- All deployed directories and files are assigned to the group `marketing`.
- All deployed directories and files are owned by the user `rroe`.

User and Group Ownership Transferal

You can switch UNIX-based user and group ownership of deployed files and directories as an option of the file permission rules. For example, upon deployment you might want to change the ownership of a set of files currently owned by the user *jdoe* to the user *rroe*. Similarly, you might want to change the group ownership from *tech_pubs* on the source host to the group *marketing* at the target host.

This feature is defined for user and groups by the `userTranslation` and `groupTranslation` elements, respectively. Both of these elements are child elements of the `permissionRules` element. The `permissionRules` element must first be specified before using these two elements. Each element has the following attributes:

- `from` — enter the existing source user or group ID (or the numerical uid or gid value assigned to a particular user or group account on the UNIX source host). For example:

  `from="jdoe"` *or*

  `from="105"`

- `to` — enter the new target user or group ID. For example:

  `to="rroe"` *or*

  `to="110"`

You must determine the appropriate user and group IDs at both the source and target hosts before you modify these attributes. You can determine these by using the `id` command at the prompt on a UNIX host. Your server will respond by displaying your user ID (UID) and group ID (GID). Enter the following command on your UNIX prompt for more details regarding the usage of this command:

```
man id
```

In the following example:

```
<permissionRules>
   <userTranslation from="jdoe" to="rroe" />
   <groupTranslation from="100" to="200" />
</permissionRules>
```

OpenDeploy will review the files in a deployment for those owned by user ID `jdoe` or group ID `100`, and assign those files a new user ownership of `rroe` and group ownership of `200` after they are deployed to the target host.

Using OpenDeploy with ACLs

Access Control Lists (ACLs) on Windows servers have the following syntax:

{ *`name`*:*`ACE`*, *`name`*:*`ACE`*, ... }

where *name* is the ACL name and *ACE* is the Access Control Entry (ACE) type.

ACL Names

The ACL name can be one of the following:

- User name
- Group name
- Domain name\user name
- Domain name\group name

ACE Types

ACEs consist of either of the following:

- Perm bits
- Standard perms

Perm Bits

Perm bits are sequences made up of one or more of the following characters:

- `R` — read
- `W` — write
- `X` — execute
- `D` — delete
- `P` — change permissions
- `O` — take ownership; equivalent to `RWX`

Standard Perms

Standard perms consists of one of the following:

- `ALL` — `RWXDPO`
- `NONE` — none
- `READ` — `RX`
- `WRITE` — `W`
- `CHANGE` — `RWXD`

In the following example:

```
    setAccess={ andre:ALL, everyone:RX }
```

OpenDeploy would remove the existing ACL and grant the user *andre* full access and the group *everyone* read access to the specified files.

In the following example:

```
    changeAccess={ chris:ALL, everyone:RX }
```

would remove any existing ACEs for *chris* and *everyone*, and grant *chris* full access and the group *everyone* read access to the specified files. Any other existing ACEs would remain unchanged.

Ignoring and Preserving ACLs

The following table describes what OpenDeploy does when the `comparisonRules` element's `ignoreAcls` attribute interacts with the `transferRules` element's `preserveAcls` attribute.

	`preserveAcls="no"`	`preserveAcls="yes"`
`ignoreAcls="no"`	Deployment includes ACLs for comparison but does not deploy the ACLs.	Deployment includes ACLs for comparison and does deploy the ACLs.
`ignoreAcls="yes"`	Deployment does not include ACLs for comparison and does not deploy ACLs.	Deployment does not include ACLs for comparison but does deploy the ACLs.

See "File Comparison Rules" on page 184 for more information on configuring the `ignoreAcls` attribute, and "File Transfer Rules" on page 186 for more information on configuring the `preserveACLs` attribute.

Enabling UNIX-Based Deployments When Extended ACLs Are Present

Deployments running on UNIX hosts where extended ACLs are present (DFS, AFS, and so forth) might fail. When deploying files between OpenDeploy UNIX hosts, OpenDeploy attempts, by default, to replicate the ACLs from the source to the target. Imposing specific user-group ownerships is done through the `permissionRules` element's `user` and `group` attributes.

On some UNIX file systems, the OpenDeploy process may be prevented from setting ACLs on deployed files and directories, which causes the deployments to fail.This is caused by extended security mechanisms introduced by file systems such as AFS and DFS.

In response to this limitation, OpenDeploy supports skipping the set ACL operations. With this functionality, the deployed items take on the ownership of the running OpenDeploy process.

To use the enhanced functionality, configure the `permissionRules` element in the deployment configuration in the following manner:

```
<permissionRules user="_iwod_user_" group="_iwod_group_" />
```

When the OpenDeploy server sees these special keywords, the set ACLs operations will be skipped. For example:

```
<target useReplicationFarm="MYFARMNAME">
   <comparisonRules dateDifferent="yes" />
   <permissionRules user="_iwod_user_" group="_iwod_group_" />
   <targetFilesystem area="/tmp/deploydir" />
</target>
```

This causes OpenDeploy to deploy files to the target on the receiver if the date is different between sender file and receiver file, and not apply the original ownership to the deployed items. The deployed items have ownership of the OpenDeploy process.

Deploying Symbolic Links

By default, a symbolic link will be transferred intact and will point to the same relative or absolute path on the target side that it was pointing to on the source side. OpenDeploy will not deploy the actual file itself, nor will it validate the link on the target side to ensure the pointed-to files reside on the target host. However, you can elect to have OpenDeploy deploy the actual pointed-to files by enabling the follow links feature. This feature is not available with OpenDeploy running on Windows.

In the following example, *foo* is a link that points to the file `/etc/reports.txt`. If you enter the following command at the prompt:

```
ls -l foo
```

the return would be

```
foo -> /etc/reports.txt
```

If *foo* is moved as part of a deployment with the follow links feature enabled at the source host side, the deployment would find the file `/etc/reports.txt` and deploy it to the target as a new version of *foo,* replacing the one already there. This feature is useful if the source host area contains links, but the files they point to reside outside of the area and would otherwise not be included in the deployment.

Source-Side

If you want to move the items pointed to by links contained on the source host area, you can enable the follow links feature with the `followLinks` attribute of the `sourceTransferRules` element. For example:

```
<sourceTransferRules followLinks="yes" />
```

Target-Side

On the target side, if you want to replace whatever the link is pointing to with what was deployed, you can enable the follow links feature in the `followLinks` attribute of the `transferRules` element. For example:

```
<transferRules followLinks="yes" />
```

Parameter Substitution

Parameter substitution is a feature that allows you to format attributes as parameters whose values you can specify on a deployment-specific basis using `iwodstart` command-line tool.You can configure any attribute value in a deployment configuration file for parameter substitution. Each time you start a deployment using `iwodstart`, you indicate the value of each attribute configured for parameter substitution. Different instances of the same deployment can have different attribute values.The parameter substitution feature turns a deployment configuration file into a template that you can customize each time you start the deployment.

To use parameter substitution, you must determine which attributes to specify as parameters, and modify them in the appropriate deployment configuration. Parameters use the following syntax:

```
$parameter^
```

where *`parameter`* is some value you will specify in conjunction with the `iwodstart` command-line tool. For example, if you wanted to designate the `area` attribute of the `sourceFilesystem` element of a particular deployment configuration as parameter `srcarea`, you would give that attribute the following value:

```
area="$srcarea^"
```

Every parameter entered into the deployment configuration file must include the dollar sign ("$") at the beginning and the carat ("^") at the end. Otherwise, you are free to name a parameter anything you want. However, each parameter name must be unique within that deployment configuration.

After your parameters are set, you can run the deployment from the command line using the `iwodstart` command-line tool. The syntax for using `iwodstart` with parameters is:

```
iwodstart deployment -k parameter=value
```

For example, if you wanted to apply the value `C:\temp` to the parameter `srcarea` in the deployment configuration file *reports*, you would enter the following command at the prompt:

```
iwodstart reports -k srcarea=C:\temp
```

Note that in the command you entered, you did not include the dollar sign or the carat in the parameter name. You also did not include the value in quotation marks. However, if either the parameter or its assigned value contained a space, then the entire combined parameter and value must be placed inside of quotation marks. For example, if the value of `srcarea` is:

```
C:\Program Files\monthly
```

then you would enter:

```
iwodstart reports -k "srcarea=C:\Program Files\monthly"
```

Multiple `-k` *`key=value`* pairs may exist on the command line, for example:

```
iwodstart reports -k src=/mysource -k trg=/mytarget
```

The `iwodstart` command can only be issued on the host where the OpenDeploy server is installed. This command can be issued by anyone regardless of whether they hold an Administrator or User role. There are no authentication or authorization checks on individuals issuing command-line tools.

See "Running Deployments from the Command Line" on page 101 for more information on that `iwodstart` command.

Use with Deployment Instances

One usage of parameter substitution is to combine this feature with specific instances of a deployment. That way, you can run multiple instances of the same deployment configuration file, but use parameter substitution to make modifications in each instance. See "Specifying a Deployment Instance" on page 103 for more information on this feature.

In the following example, there are two instances of the deployment reports being run using parameter substitution.

```
iwodstart reports -inst monthly -k "srcarea=C:\Program Files\monthly"

iwodstart reports -inst quarterly -k "srcarea=C:\Program Files\quarterly"
```

In the first instance `monthly`, the source file area is specified as `C:\Program Files\monthly` using parameter substitution. In the second instanced quarterly, the source file area is specified as `C:\Program Files\quarterly.`

Use with Scheduled Deployments

You can also apply parameter substitution to scheduled deployments. See "Applying Parameter Substitution to Scheduled Deployments" on page 234 for more information.

Deploy and Run

The *Deploy and Run* feature allows you to configure OpenDeploy to execute an external script at a specified stage of the deployment. This stage can be the deployment of a particular type or class of file or directory, or even the success of the deployment itself. Using Deploy and Run, you can configure OpenDeploy to do the following tasks automatically:

- Execute a notification script upon a failed deployment
- Run a language-checking script during deployment
- Alert you each time an executable file (with a file extension of `.exe`) is deployed

OpenDeploy supports any scripting language. The script must reside on the server where it is to be invoked. OpenDeploy will not transfer that script.

Requirements

Deploy and Run requires the following components:

- The presence of a customized script to be run.
- A directive indicating in what situation the script should be invoked:
  - Deployment of a specific file or filenames matching a certain pattern
  - Deployment [creation] of a specific directory
  - Before or after the actual deployment.
  - On the source or target side of the deployment
  - On success or failure of the deployment, or always

  Not all of these options are applicable in combination with one another.

Deploy and Run Trigger Stages

You can configure Deploy and Run scripts to be triggered on a total deployment basis (macrodeploy), or by specific task events (microdeploy) that occur in the course of the deployment, such as a connection with a particular target. Where within this cycle you want to add Deploy and Run scripts determines how Deploy and Run is configured.

Figure 38 shows those deployment- and task-level steps where you can set Deploy and Run scripts.

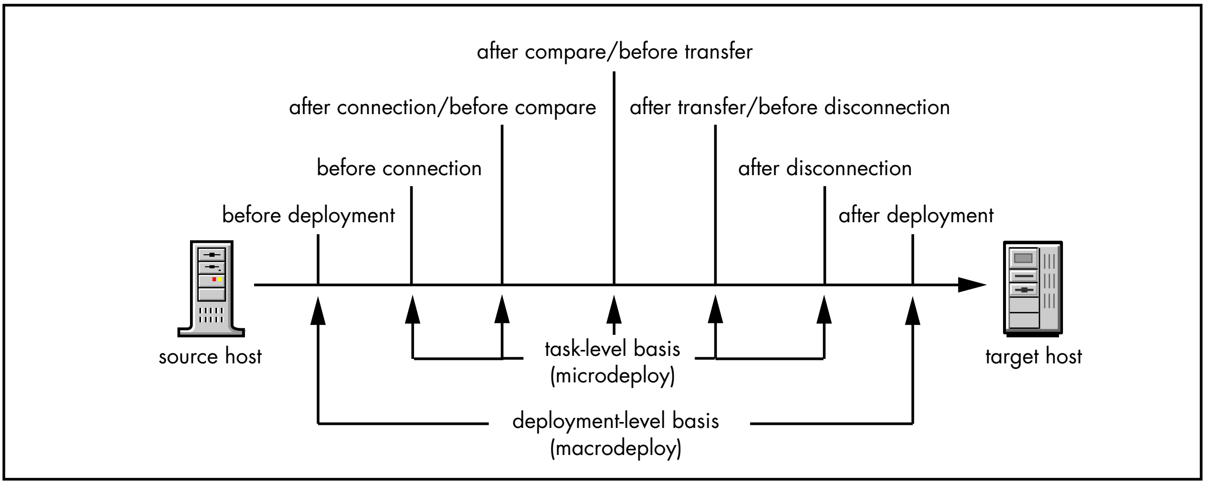


Figure 38: Deploy and Run Stages in the Deployment

Deployment-Level Triggers

Deployment-level, or *macrodeploy*, triggers are associated with the entire deployment as a single entity. They can be configured to occur either at the beginning of a deployment before the connection between the source and targets occurs, or following the completion or termination of a deployment at the time of host disconnection. They also can be configured to trigger if the deployment is successful, a failure, or under both conditions.

You configure deployment-level Deploy and Run triggers and scripts within the dnrDeploymentJob element:

```
<deployment ...>
   <dnrDeploymentJob location="source" when="after" state="success">
         <script ... />
   </dnrDeploymentJob>
   ...
</deployment>
```

The dnrDeploymentJob has the following associated attributes:

- location — indicate indicates whether the Deploy and Run script is taking place on the source or target host. This value is fixed as source.
- when — indicate whether the script should be executed before (before) or after (after) the deployment of the particular directory. There is no default value. You must specify one of the options.
- state — indicate whether the Deploy and Run script should run as a result of the success (success) or failure (failure) of the deployment, or whether it should always run in either case (always). Default value is always.

Within the dnrDeploymentJob element is the script element where the particular script that is run when triggered is specified. See "Deploy and Run Scripting" on page 211 for more information.

Task-Level Triggers

Task-level, or *microdeploy*, triggers occur within the course of the deployment between the time the initial connection is made between the source and targets, and when the disconnection between the hosts occurs. Unlike deployment-level triggers which are only associated with the complete deployment, task-level triggers are associated with a specific deployment action, such as a deployment to a particular target in a fan-out deployment, or a particular leg in a multi-tiered deployment.

Task-level triggers can be associated with the following events:

- Deployment of a particular file.
- Deployment of a particular directory.
- Running of a particular deployment definition.

Deployment-based definitions are triggered by different stages of the deployment process.

File- and directory-based Deploy and Run definitions are data-oriented triggers that run when the deployed file or directory matches a specified pattern. These types of triggers are not supported in transactional deployments.

You configure task-level Deploy and Run within the `deployNRun` element, which is a child element of the `execDeploymentTask` element. Task-level triggers are associated with a particular definition within the deployment configuration. In the deployment configuration, the particular `definition` element's name is specified by the `execDeploymentTask` element's `useDefinition` attribute value. For example, if in a deployment the following definition was configured:

```
<definition name="webfiles">
   <source>
      ...
   </source>
   <target ...>
      ...
   </target>
</definition>
```

Then any task-level Deploy and Run triggers and scripts you want to be associated to that definition and its source-target file location pairing would be configured:

```
<deployment ...>
   ...
   <execDeploymentTask useDefinition="webfiles">
      <deployNRun>
         ...
      </deployNRun>
   </execDeploymentTask>
</deployment>
```

Within the deployNRun element is the configuration for the various supported task-level triggers and scripts:

```
<deployment ...>
   <dnrDeploymentJob location="source" when="after" state="success">
         <script ... />
   </dnrDeploymentJob>
   ...
   <execDeploymentTask useDefinition="webfiles">
      <deployNRun>
         <dnrFile ...>
            <script ... />
         </dnrFile>
         <dnrDir ...>
            <script ... />
         </dnrDir>
         <dnrDeployment ...>
            <script ... />
         </dnrDeployment>
      </deployNRun>
   </execDeploymentTask>
</deployment>
```

Here is a list of child elements associated with the deployNRun element:

- dnrDeployment — specifies under what conditions a deployment itself can trigger a Deploy and Run script.
- dnrFile — specifies under what conditions deployed files can trigger a Deploy and Run script.
- dnrDir — specifies under what conditions deployed directories can trigger a Deploy and Run script.

You can configure any number of these triggers in any combination with each other. The following sections describe each type of trigger in detail.

Deployment-Based

You can configure OpenDeploy to begin a Deploy and Run script when the deployment configuration itself is run. This type of Deploy and Run is known as being *deployment-based*. You enable this feature by defining the `dnrDeployment` element in your Deploy and Run configuration. The `dnrDeployment` element has the following associated attributes:

- `location` — indicate whether the Deploy and Run script is taking place on the source (`source`) or target (`target`) host. There is no default value. You must specify one of the options.

- `when` — indicate whether the script should be executed before (`before`) or after (`after`) the deployment occurs. There is no default value. You must specify one of the options.

  Evaluation of the `area` and `previousArea` attribute values in TeamSite comparison deployments with respect to the latest and next-to-latest editions, occurs *before* the running of any Deploy and Run scripts.

- `triggerPoint` — indicate whether the Deploy and Run script should run at the time or connect between the source and host (`connect`), during the transfer of content (`transfer`), or at the time of disconnect (`disconnect`). If no value is specified, OpenDeploy defaults to certain settings depending on the specified value of the when attribute.

  – `when="before"` — trigger occurs after the source-target connect occurs.

  – `when="after"` — trigger occurs after the content transfer occurs.

  See “Selecting the Task-Level Deployment Stage” on page 207 for more information.

- `state` — indicate whether the Deploy and Run script should run as a result of the success (`success`) or failure (`failure`) of the deployment, or whether it should always run in either case (`always`). Default value is `always`.

  If the `when` attribute is set to `before`, then the `state` attribute is implicitly set to `always`, because there has been no deployment yet to determine success or failure.

Selecting the Task-Level Deployment Stage

There are several deployment stages in a task where you can assign Deploy and Run triggers. You can specify the stage by the pairing of the `when` and `triggerPoint` attribute values:

- At the beginning of the task:

  ```
  when="before" triggerPoint="connect"
  ```

- Following the source-target connection:

  ```
  when="after" triggerPoint="connect"
  ```

  As an option, you can omit the `triggerPoint` attribute from this configuration and still have the same result. You might want to configure the trigger in this way if you want to retain backwards deployment configuration compatibility with previous OpenDeploy releases. Otherwise, it is recommended you retain the `triggerPoint` attribute in the configuration.

- At the beginning of the transfer of content:

  ```
  when="before" triggerPoint="transfer"
  ```

  If you use this configuration, you must also have the deployment manifest feature enabled within the base server or receiver host configuration file. Refer to "Specifying the Deployment Information Stream Format" on page 67 in the *OpenDeploy Installation and Reference Guide* for more information.

- At the ending of the transfer of content:

  `when="after" triggerPoint="transfer"` *or*

  ```
  when="before" triggerPoint="disconnect"
  ```

  As an option, you can omit the `triggerPoint` attribute from this configuration and still have the same result. You might want to configure the trigger in this way if you want to retain backwards deployment configuration compatibility with previous OpenDeploy releases. Otherwise, it is recommended you retain the `triggerPoint` attribute in the configuration.

- At the end of the task:

  ```
  when="after" triggerPoint="disconnect"
  ```

No other combination of `when` and `triggerPoint` values is supported (such as `when="before" triggerPoint="connect"`). If any non-supported combination of these values is present in the deployment configuration, then the Deploy and Run script will not trigger when that deployment is run.

In the following example:

```
<dnrDeployment
   location="source"
   when="after"
   triggerPoint="transfer"
   state="failure"
   >
   ...
</dnrDeployment>
```

the Deploy and Run script triggers when this particular deployment configuration is run. The script originates at the source host, and is executed at the conclusion of the transfer of content of a failed deployment.

File-Based

You can configure OpenDeploy to begin a Deploy and Run script when a certain type or class of files are deployed. This type of Deploy and Run is known as being *file-based*. It is supported only for non-transactional deployments. You enable this feature by defining the `dnrFile` element in the Deploy and Run configuration. File-based Deploy and Run is not available for use with transactional deployments.

The `dnrFile` has the following associated attributes:

- `location` — indicate where the Deploy and Run script is taking place. The only currently supported option is `target`.
- `when` — indicate whether the script should be executed before (`before`) or after (`after`) the deployment of the particular file. There is no default value. You must specify one of the options.
- `state` — indicate whether the Deploy and Run script should run as a result of the success (`success`) or failure (`failure`) of the deployment, or whether it should always run in either case (`always`) . Default value is `always`.

If the when attribute is set to before, then the state attribute is implicitly set to always, because there has been no deployment yet to determine success or failure.

- mask — enter the regular expression to be matched against filenames to determine if the script will be executed. If you include file path separators in your mask value, you must use the path syntax supported by your OpenDeploy server.

  In the following example:

  ```
  mask=".*\.html$"
  ```

  any file with the file extension .html in the specified path will trigger the script defined within the Deploy and Run configuration.

In the following example:

```
<dnrFile
   location="target"
   when="before"
   state="always"
   mask=".*\.exe$"
   >
   ...
</dnrFile>
```

the Deploy and Run script, when triggered, will be located on the target host. It will occur before a file matching the value of the mask attribute is deployed. OpenDeploy will trigger the script whether the deployment is successful or not. (If the when attribute is set to before, the state attribute is implicitly set to always because there has been no deployment yet to determine success or failure.) The mask attribute value .*\.exe$ indicates that the script will start each time a file with the extension .exe is deployed.

If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

Directory-Based

You can configure OpenDeploy to begin a Deploy and Run script when a certain type or class of directories are deployed. This type of Deploy and Run is known as being *directory-based*. It is supported only for non-transactional deployments. You enable this feature by defining the `dnrDir` element in the Deploy and Run configuration. Directory-based Deploy and Run is not available for use with transactional deployments.

The `dnrDir` has the following associated attributes:

- `location` — indicate where the Deploy and Run script is taking place. The only currently supported option is `target`.
- `when` — indicate whether the script should be executed before (`before`) or after (`after`) the deployment of the particular directory. There is no default value. You must specify one of the options.
- `state` — indicate whether the Deploy and Run script should run as a result of the success (`success`) or failure (`failure`) of the deployment, or whether it should always run in either case (`always`). Default value is `always`.

  If the `when` attribute is set to `before`, then the `state` attribute is implicitly set to `always`, because there has been no deployment yet to determine success or failure.
- `mask` — enter the regular expression specifying the directories that, when deployed, will trigger the script. You must use the path syntax supported by your OpenDeploy server. The script is triggered *only* when the directory itself is deployed on the target host. Deploying a file that resides within the directory will *not* trigger the script.

  In the following example:

  ```
  mask="cgi-bin$"
  ```

  any directory named `cgi-bin` that is deployed will trigger the script.

In the following example:

```
<dnrDir
   location="target"
   when="after"
   state="success"
   mask="Temp$"
   >
   ...
</dnrDir>
```

the Deploy and Run script, when triggered, will be located on the target host. It will occur after a directory matching the value of the `mask` attribute is deployed, and only if the deployment is successful. The `mask` attribute value `Temp$` indicates that the script will start each time a directory with the name `Temp` is deployed.

If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

Deploy and Run Scripting

The Deploy and Run script is defined by the `script` element. Once the script is in place, the `script` element will integrate it into the Deploy and Run configuration. The `script` element has the following attributes:

- `cmd` — enter the full path to the command which OpenDeploy should run, if triggered, as well as any accompanying flags or options. If you specify the `where` attribute (see below) you may be able to use the value "`./`" on UNIX, or no path specification on Windows. You can also specify an executable invocation line. For example:

  `cmd="C:\bin\email_to_admin.bat -user jdoe@interwoven.com"` *or*

  `cmd="/bin/mail jdoe@interwoven.com < /tmp/message.txt"`

If the command you are going to run requires a scripting engine, the scripting engine must be on the PATH of the user (or system, on Windows) who will be running the script or specified with a full path). For example:

```
cmd="/bin/sh /usr/local/bin/email_to_admin.sh -u jdoe@interwoven.com"
```
or

```
cmd="/usr/local/bin/iwperl /path/to/script.pl"
```

- `where` — enter the location to where OpenDeploy must go before executing the script. For example:

  ```
  where="/tmp"
  ```
  or

  ```
  where="C:\temp"
  ```

  You must use the path syntax supported by your OpenDeploy server. Forward slashes ("/") can be used for either Windows or UNIX hosts, while backslashes ("\") are only permitted on Windows hosts.

  The `where` attribute is optional. If you do not specify a value, the process takes place in the root directory.

- `as` (UNIX only) — enter a different user name or user ID under which you will run the script. This option allows you to run the script as a different user. In the following example:

  ```
  as="rroe"
  ```
  or

  ```
  as="110"
  ```

  you can run the script as `rroe` or its user ID `110` rather than your regular user name. By default, the script runs as the user who invokes OpenDeploy. The user who will need to be root for most purposes.

- `async` — indicate whether or not to run the script asynchronously. Exercise caution when using this mode, as it could cause many scripts to be run simultaneously. The output from scripts run asynchronously is not captured. Default value is `no`.

In the following example:

```
<dnrDeployment
   location="source"
   when="after"
   triggerPoint="connect"
   state="always"
   >
   <script
      cmd="/bin/sh /usr/local/bin/email_to_admin.sh"
      where="/tmp"
      async="yes"
      />
</dnrDeployment>
```

the Deploy and Run configuration will trigger the `/bin/sh /usr/local/bin/email_to_admin.sh` script (which sends deployment confirmation email messages to the OpenDeploy administrator) after the deployment has completed, whether successful or not. The script is also allowed to run asynchronously.

The following configuration sample shows Deploy and Run configured both at the deployment level and at the task level. The definition with which the task-level Deploy and Run configuration is associated is also displayed.

```
<definition name="webfiles">
   <source>
      <sourceFilesystem area="C:\website\files">
         <pathSpecification>
            <path name="monthly" />
         </pathSpecification>
      </sourceFilesystem>
   </source>
   <target useReplicationFarm="WebServers">
      <targetFilesystem area="D:\website\files" />
   </target>
</definition>
```

```
<deployment transactional="no">
   <dnrDeploymentJob location="source" when="after"
      triggerPoint="connect" state="success">
            <script cmd="C:\default\main\dev\scripts" as="webmaster"
               where="C:\temp" async="yes" />
   </dnrDeploymentJob>
   <execDeploymentTask useDefinition="webfiles>
      <deployNRun>
         <dnrFile location="target" when="before" state="success"
            mask="\.txt$">
            <script cmd="email_to_admin.bat" as="webmaster" where="C:\temp"
               async="no" />
         </dnrFile>
         <dnrDir location="target" when="after" state="failure" mask="exes">
            <script cmd="mail jdoe@interwoven.com < message.txt"
               as="webmaster" where="C:\temp" async="no" />
         </dnrDir>
         <dnrDeployment location="source" when="after" triggerPoint=""
            state="success">
            <script cmd="C:\default\main\dev\scripts" as="webmaster"
               where="C:\temp" async="yes" />
         </dnrDeployment>
      </deployNRun>
   </execDeploymentTask>
</deployment>
```

Deploy and Run Usage of the Deployment Information Stream

OpenDeploy generates an internal list of path items deployed or to be deployed each time a deployment is run. This data can be streamed into a Deploy and Run script, which you can configure to produce its own log file by re-piping `STDIN` to another file. After the stream is consumed by the Deploy and Run script, you can manipulate it to meet your needs.

The following steps take place whenever Deploy and Run calls a script:

1. `stdin` of the spawned Deploy and Run process is set to receive an XML representation of the OpenDeploy in-memory log in its current state.
2. The script executes and any results can be written to `stdout`.
3. The receiver XML log is transferred to the sender.

In this manner, future scripts can parse the output of past scripts. For example, a script might extract information about which files were deleted during the last deployment.

In Figure 39, one Deploy and Run script is configured to run before a deployment, and a second Deploy and Run script is configured to run after the deployment. The second script is responsible for transferring information from `STDIN` to a separate log file (`foo.log`).

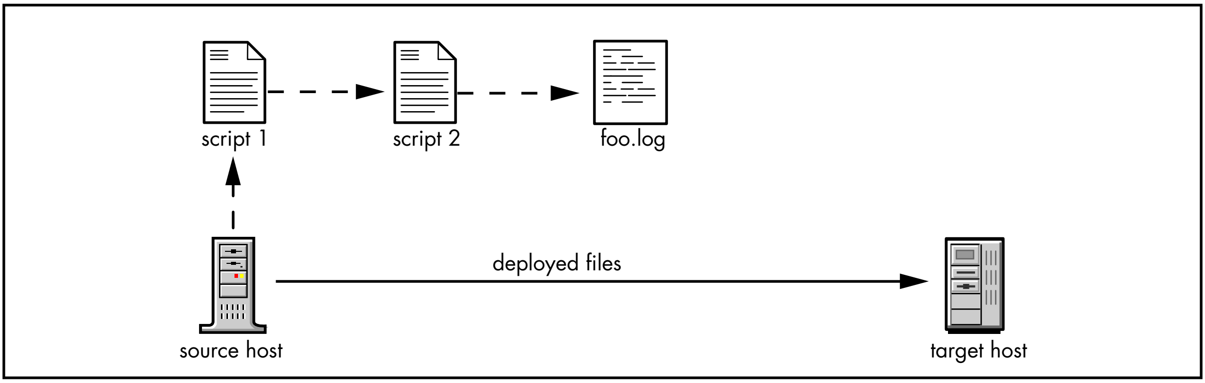


Figure 39: Deploy and Run Logging

The first Deploy and Run script (`script 1`) has the option to read from `STDIN` and extract details regarding the deployment. This script can then send output to `STDOUT` which will be captured by OpenDeploy's in-memory log. The second Deploy and Run script (`script 2`) parses `STDIN` which is in XML format. This script can retrieve details regarding the deployment, and the output from the first Deploy and Run script. These details can then be written to a log file (`foo.log`).

Use the Perl module `IWXML.pm` to process either information stream format from a Perl program. This module resides in the following location:

```
od-home/solutions/perl
```

Information Stream Output Formats

This release of OpenDeploy contains a newer method of capturing this streamed information into a new *manifest* format. Older releases of OpenDeploy used a legacy *log* format. Both methods are supported.

Refer to “Specifying the Deployment Information Stream Format” on page 67 in the *OpenDeploy Installation and Reference Guide* for information on configuring these formats in your host configuration file.

The following is an example of the information stream outputted in the manifest format:

```
<iwodManifest type="transfer">
<profile  srcPlatform="UNIX" srcDir="/space/ODadvanced/OpenDeployNG/solutions/
perl" trgPlatform="UNIX" trgDir="/space/ODadvanced/OpenDeployNG/tmp/viewstream"/>
<item path="deletefile.txt" type="FILE" reason="missing-in-src" action="DELETE"
status="SKIPPED" statusDetail="" srcDir="/space/ODadvanced/OpenDeployNG/
solutions/perl" trgDir="/space/ODadvanced/OpenDeployNG/tmp/viewstream"/>
<item path="modfile.txt" type="FILE" reason="src-is-newer" action="UPDATE"
status="COMPLETED" statusDetail="" srcDir="/space/ODadvanced/OpenDeployNG/
solutions/perl" trgDir="/space/ODadvanced/OpenDeployNG/tmp/viewstream"/>
<item path="newfile.txt" type="FILE" reason="missing-in-dest" action="NEW"
status="COMPLETED" statusDetail="" srcDir="/space/ODadvanced/OpenDeployNG/
solutions/perl" trgDir="/space/ODadvanced/OpenDeployNG/tmp/viewstream"/>
</iwodManifest>
```

The following is an excerpt of the same information stream outputted in the log format:

```
...
<log_element  target="" action="0" date="1043364928" result="3"
response="directive[get ./new.txt]" />
<log_element  target="" action="0" date="1043364928" result="1"
response="Receiving item(./new.txt)" />
<log_element  target="" action="0" date="1043364928" result="3" response="dir for
tempfile [/space/ODadvanced/OpenDeployNG/tmp/viewstream]" />
<log_element  target="" action="0" date="1043364928" result="3" response="--
Processing file(/space/ODadvanced/OpenDeployNG/tmp/viewstream/new.txt)" />
<log_element  target="" action="0" date="1043364928" result="3" response="file
created[/space/ODadvanced/OpenDeployNG/tmp/viewstream/new.txt.iwtmp]" />
<log_element  target="" action="0" date="1043364928" result="3" response="Renamed
[/space/ODadvanced/OpenDeployNG/tmp/viewstream/new.txt.iwtmp] to [/space/
ODadvanced/OpenDeployNG/tmp/viewstream/new.txt.iwnew]" />
<log_element  target="/space/ODadvanced/OpenDeployNG/tmp/viewstream/new.txt"
action="1" date="1043364928" result="0" response="" />
<log_element  target="" action="0" date="1043364928" result="3"
response="directive[reason missing-in-dest]" />
...
```

The manifest format provides a more concise presentation that is easier to read and understand.

For DTD information on manifest and log formats, refer to Chapter 8, "OpenDeploy Manifest DTD" on page 197 and Chapter 9, "Information Stream Log Format DTD" on page 203, respectively, in the *OpenDeploy Installation and Reference Guide*.

Communicating Status to OpenDeploy

A DNR script can send a return code to OpenDeploy by printing the following:

```
<response code="-2"/>\n
```

to `STDOUT`. A value of `-2` indicates that the result of the action was to signal the deployment to do a hard abort in which the deployment stops immediately. If the deployment was transactional, the files on the production server will be returned to their original state.

Disabling Deploy and Run Executions on the Target Host

You can disable Deploy and Run executions on the target host of a deployment where Deploy and Run is configured in the host's base server or receiver configuration file.This ability does not have any effect on the sending host's Deploy and Run executions specified in the deployment configuration.

To disable the running of Deploy and Run executions on the target host, you must add the `restrictDnr` element and its `allowDnrExecution` attribute to the `deployServerConfiguration` element in the target host's base server or receiver configuration file. For example:

```
<deployServerConfiguration>
    ...
       <restrictDnr allowDnrExecution="no" />
    ...
</deployServerConfiguration>
```

If you want to disable Deploy and Run executions on the target host, specify the `allowDnrExecution` attribute value as `no`. To run Deploy and Run executions on the target host without restrictions, specify the value as `yes`. By default, if the `restrictDnr` element is not present in the configuration, the ability to execute Deploy and Run on the target host will not be prevented.

Deploying to a Package File Using Deploy and Run

If it is impossible or impractical to deploy files directly to your target hosts, perhaps because of a firewall or some other obstruction, OpenDeploy can deploy files into a package file on another host, or even the source host itself. You can transport the file to the target hosts using an approved means, and install the deployed files directly on the targets. OpenDeploy uses the Deploy and Run feature as the basis for creating package files.

In Figure 40, a direct transmission of files between the source host and the target host is not possible. The OpenDeploy administrator configures a deployment to write the deployed files to a package file, such as a `.tar` or `.zip` file, on the source host. That package file is then copied to a transportable medium, such as a tape, and is subsequently manually installed on the target host.

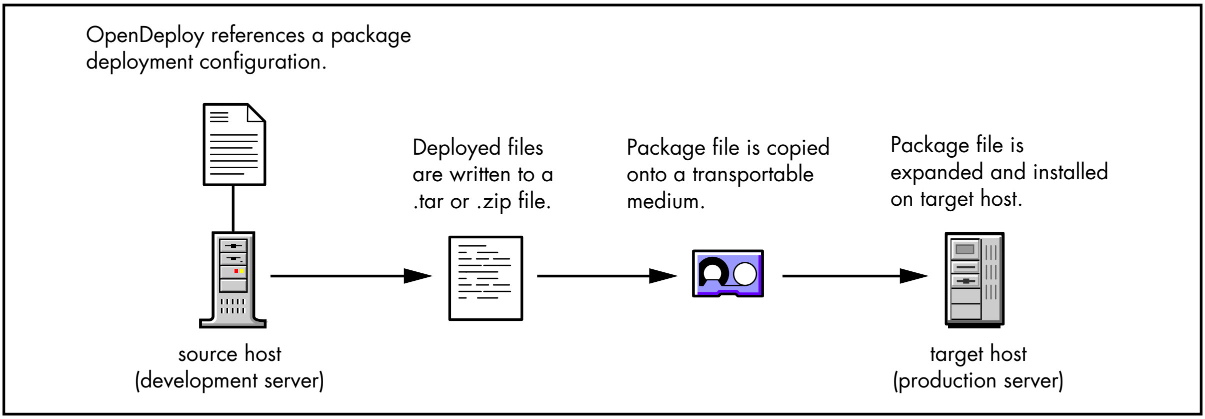

OpenDeploy references a package deployment configuration.
Deployed files are written to a .tar or .zip file.
Package file is copied onto a transportable medium.
Package file is expanded and installed on target host.
source host (development server)
target host (production server)

Figure 40: Package Deployment

To create a package file, follow these steps:

1. Configure a deployment that deploys files to the source host itself. This involves the following tasks:
   – Specifying your source host in the nodes configuration file.
   – Adding your host and target directory to the `allowedHosts` and `allowedDirectories` elements in your base server configuration file.

2. Configure the deployment with a Deploy and Run script that writes the deployed files to a package file after a successful deployment. On an OpenDeploy host running on a UNIX host, this would appear in your deployment configuration as the following:

   ```
   <dnrDeployment location="target" when="after" state="success">
      <script cmd="tar cvf package_file.tar target_area" async="no" />
   </dnrDeployment>
   ```

   where *package_file*.tar is the name of the package file and *target_area* is the path to the location where the deployed files will reside after completing the deployment.

On an OpenDeploy host running on Windows, you would substitute a Windows packaging command for the cmd attribute.

Secure Invocation of External Applications on UNIX

The Deploy and Run scripting facility has been enhanced to launch external applications on UNIX servers using the deployment user's ID as the process owner. This applies to both sender- and receiver-side Deploy and Run invocations. This feature helps prevent a sender from launching potentially harmful operations that the user is not permitted to perform on sending and receiving systems.

Although this feature applies only to deployments running on UNIX hosts, deployments can be initiated from OpenDeploy running on either UNIX or Windows hosts. By default, the Deploy and Run script will run as the user who invoked the deployment if the user attribute is unspecified in the Deploy and Run script element.

Utilizing the Delivery Adapter Framework

The Delivery Adapter Framework enables the creation of application-specific adapters for extending the content delivery capabilities of OpenDeploy. This allows you to extend the reach of your content distribution network to specialized devices and protocols, such as edge or cache servers.

The Delivery Adapter Framework allows the referencing of a user-created Java callout in the deployment configuration. After content is deployed to a target host running the base server software (target hosts running the receiver software cannot use this feature), the Java class that implements the adapter is invoked with a manifest of files and any other adapter properties that are specified in the deployment configuration.

When the adapter is invoked after the content is received, it has a handle to the manifest file (which lists the items deployed or deleted on the target side). The adapter can walk through the manifest to perform whatever operations are deemed appropriate by the developer of the adapter. Adapter developers can parse through the manifest file and take the necessary action based on their requirement.

An example of a Java adapter is provided in the following location:

```
od-home/solutions/adapter/example
```

Adapters are invoked on the target side after the deployment. Adapters cannot be invoked on the source side.

Adding Adapters to the Deployment Configuration

Utilization of the Delivery Adapter Framework is defined in the `odAdapterSet` element in the deployment configuration. The `odAdapterSet` element is a child element of the `target` element and acts as a container for individual `odAdapter` elements:

```
<target ...>
   <odAdapterSet>
      <odAdapter ... />
      <odAdapter ... />
      ...
   </odAdapterSet>
</target>
```

Each `odAdapter` element contains the following required attributes:

- `name` — specify the name of the Java class that implements the adapter. For example:

  ```
  name="com.interwoven.od.adapter.netcache.IWODNetCache"
  ```

- `parameter` — specify the parameter string for the adapter. The developer of the adapter is responsible for defining the meaning and syntax of the parameter string. For example:

  `parameter="xyz"` *or*

  `parameter="`*`/my-dir/my-parm-file`*`"`

  If there are no associated parameters for the specified Java class, you can give the `parameter` attribute a null value. For example:

  ```
  parameter=""
  ```

Writing Java Adapters

The following instructions are included to assist you in writing your own Java adapters for use in the Delivery Adapter Framework. A `README` file containing these instructions resides in the following location:

`od-home``/solutions/adapter/README`

To write a Java adapter, follow these steps:

1. Add `baseadapter.jar` to your classpath. This jar file contains the base implementation of the adapter.

   Your class should be derived from `IWODAdapter` (contained in `basedadapter.jar`) and it should implement the following methods:

   – A constructor that has no parameters, for example: `public AdapterExample()`

   – A run method taking no parameters, for example: `run()`

   The method signature would be: `public void run()`

2. Add your own adapter implementation in your new class and compile it into a class file.

3. Package your Java class files into a `.jar` file and name it `odadapter.jar`.

4. Copy `odadapter.jar` to the following location:

   `od-home/userlib`

5. Add the `odAdapter` element in your deployment configuration file. Supply your class name (which is derived from `IWODAdapter`) as the `name` attribute of this `odAdapter` element. See “Adding Adapters to the Deployment Configuration” on page 220 for more information.

6. Start the deployment. The adapter implementation is invoked after the deployment.

The files and configuration instructions to use the Delivery Adapter Framework for specific adapters are included and described in Appendix A, “Java Adapters” on page 365.

Chapter 5

Scheduled Deployments

You can schedule a deployment to take place any time day or night.You can schedule the deployment to run on a one-time only basis, or recurrently on intervals from a few minutes to monthly. Scheduling deployments frees individuals from having to manually start a deployment.

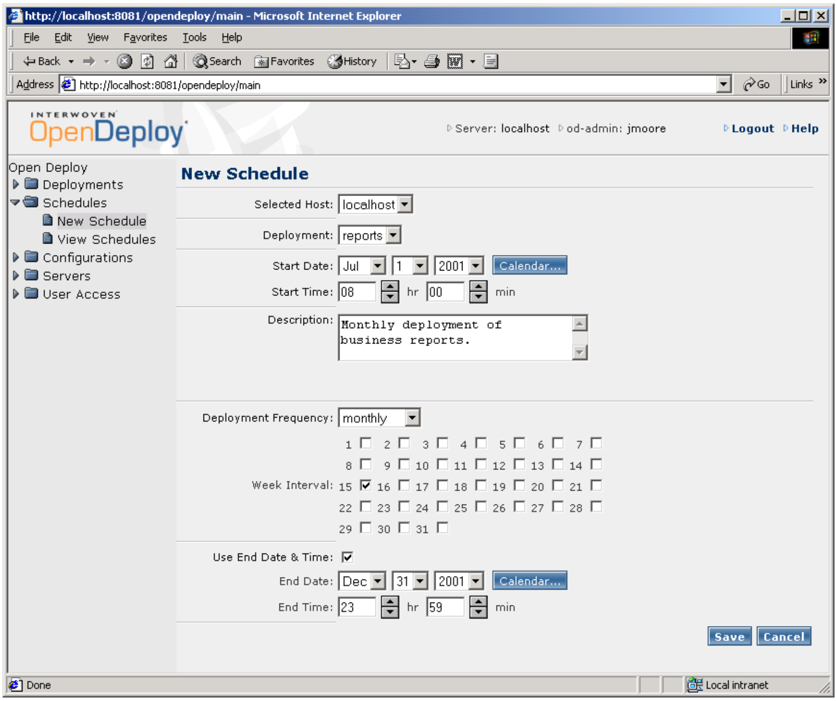
You can schedule deployment to take place at low network traffic periods such as evenings and weekends when they will not interfere with other tasks. You can also schedule a deployment to take place in conjunction with other events, such as a product announcement.

Any individual holding an Administrator role on the OpenDeploy server can schedule any deployment on that host. Individuals with User accounts on an OpenDeploy server can schedule those deployments assigned to them. Individuals holding either an Administrator or User role can view all schedules.

You can schedule deployments using the user interface or from the command line using command-line tools.

Scheduling from the User Interface

You can create, edit, delete, and view deployment schedules in the OpenDeploy user interface. Creating and editing schedules is performed in the New Schedule window (Figure 41).


http://localhost:8081/opendeploy/main - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8081/opendeploy/main
Go Links
INTERWOVEN OpenDeploy
Logout Help
Open Deploy
Deployments
Schedules
New Schedule
View Schedules
Configurations
Servers
User Access
Calendar...
Save Cancel
Done
Local intranet

Server: localhost od-admin: jmoore

New Schedule

Selected Host: localhost

Deployment: reports

Start Date: Jul 1 2001

Start Time: 08 hr 00 min

Description: Monthly deployment of business reports.

Deployment Frequency: monthly

Week Interval:
1 □ 2 □ 3 □ 4 □ 5 □ 6 □ 7 □
8 □ 9 □ 10 □ 11 □ 12 □ 13 □ 14 □
15 ☑ 16 □ 17 □ 18 □ 19 □ 20 □ 21 □
22 □ 23 □ 24 □ 25 □ 26 □ 27 □ 28 □
29 □ 30 □ 31 □

Use End Date & Time: ☑

End Date: Dec 31 2001

End Time: 23 hr 59 min

Figure 41: New Schedule Window

Here you can specify the time and date the deployment will start. If you want it to occur more than once on a regular basis, such as daily or weekly, you can select that as well. Depending on the frequency level you assign to the scheduled deployment, the New Schedule window will prompt you for additional scheduling information. You can also specify an end date when the schedule is no longer in effect.

Scheduling Deployments

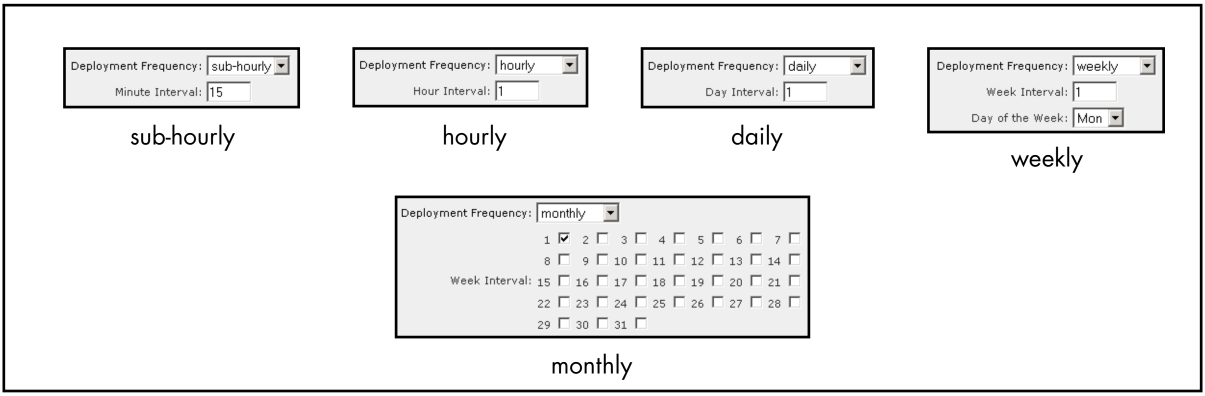
To schedule a deployment, follow these steps:

1. Select **Schedules > New Schedule** to display the New Schedule window.
2. Select the OpenDeploy server whose deployments you want to schedule from the **Selected Host** list.
3. Select the deployment you want to schedule from the **Deployment** list.
4. Select the month, day, and year on which you want to the deployment to start from the **Start Date** lists. You can also click **Calendar** to display a pop-up calendar window. Select the date in this window, and it will automatically be placed in the **Start Date** lists.
5. Select the hour and minute on which you want the deployment to start from the **Start Time** lists. Use the 24-hour clock system, such as `13` to indicate 1 pm.
6. Enter a description of the deployment in the **Description** box. For example:

   ```
   This is a deployment that updates all our product pages nightly.
   ```

7. Select one of the following options from the **Deployment Frequency** list:
   - **Once** — select if the deployment is not recurring.
   - **Sub-hourly** — select to enable deployments recurring in a fixed number of minutes. The **Sub-Hourly** section appears at the bottom on the window (Figure 42). Enter the interval in minutes between deployments in the **Minute Interval** box.
   - **Hourly** — select to enable deployments recurring in a fixed number of hours. The **Hourly** section appears at the bottom on the window (Figure 42). Enter the interval in hours between deployments in the **Hour Interval** box.
   - **Daily** — select to enable deployment recurring in a fixed number of days. The **Daily** section appears at the bottom on the window (Figure 42). Enter the interval in days between deployments in the **Day Interval** box.
   - **Weekly** — select to enable deployment recurring in a fixed number of weeks, and on the same day. The **Weekly** section appears at the bottom of the window (Figure 42). Enter the interval in weeks between deployments in the **Week Interval** box. Select the day of the week the deployment will occur in the **Day of the Week** list.

– **Monthly** — select to enable deployments recurring every month on the same date. The **Monthly** section appears, containing a 31 day calendar (Figure 42). Check each date that the monthly deployment will occur. If you select a date that does not occur every month, for example "31," then that deployment will not occur until the next month that includes that date. A date of "31" would skip June, but take place in July.



Deployment Frequency: sub-hourly
Minute Interval: 15

sub-hourly

Deployment Frequency: hourly
Hour Interval: 1

hourly

Deployment Frequency: daily
Day Interval: 1

daily

Deployment Frequency: weekly
Week Interval: 1
Day of the Week: Mon

weekly

Deployment Frequency: monthly

Week Interval:

1 ☑ 2 □ 3 □ 4 □ 5 □ 6 □ 7 □
8 □ 9 □ 10 □ 11 □ 12 □ 13 □ 14 □
15 □ 16 □ 17 □ 18 □ 19 □ 20 □ 21 □
22 □ 23 □ 24 □ 25 □ 26 □ 27 □ 28 □
29 □ 30 □ 31 □

monthly

Figure 42: New Schedules Frequency Features

If you selected any deployment frequency option other than **Once**, continue to the next step. Otherwise, click **Save** to complete the schedule.

8. Check the **Use End Date & Time** box if you want to designate an end date for the recurring deployments. If you do not check this box, the recurring deployments will take place indefinitely.

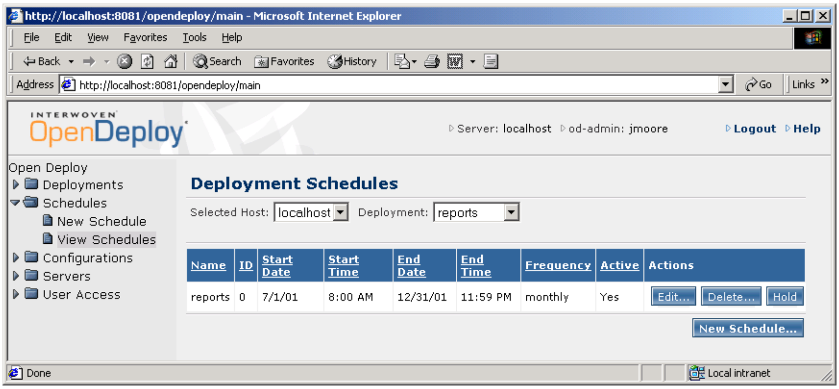
9. Select the month, day, and year on which you want to the deployment to end from the **End Date** lists. You can also click **Calendar** to display a pop-up calendar window. Select the date in this window, and it will automatically be placed in the **End Date** lists.

10. Select the hour and minute on which you want the recurring deployment to end from the **End Time** lists.

11. Click **Save** to complete the new schedule. The View Schedules window appears, displaying the new schedule you just created along with the other scheduled deployments.

Viewing Deployment Configuration Schedules

Each time you add a schedule, that schedule is displayed in the Deployment Schedules window (Figure 43).


http://localhost:8081/opendeploy/main - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8081/opendeploy/main
Go Links
INTERWOVEN OpenDeploy
Server: localhost od-admin: jmoore
Logout Help
Open Deploy
Deployments
Schedules
New Schedule
View Schedules
Configurations
Servers
User Access
Done
Local intranet

Deployment Schedules

Selected Host: localhost Deployment: reports

Name	ID	Start Date	Start Time	End Date	End Time	Frequency	Active	Actions
reports	0	7/1/01	8:00 AM	12/31/01	11:59 PM	monthly	Yes	Edit... Delete... Hold

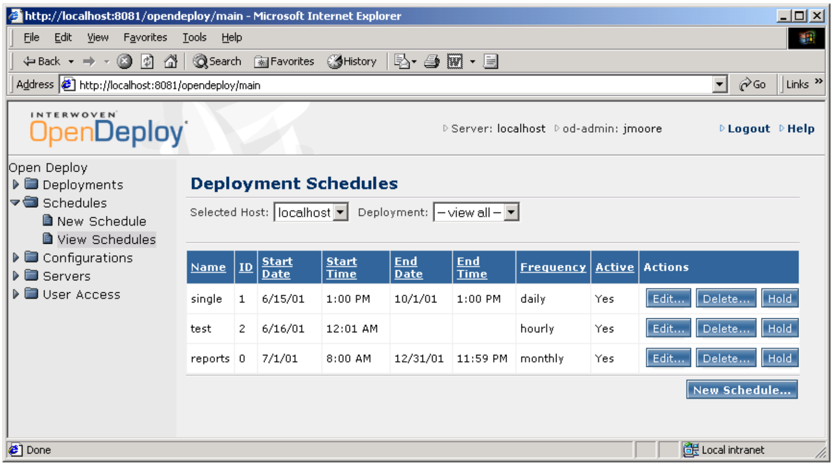
New Schedule...

Figure 43: Deployment Schedules Window

This window displays the following information regarding your scheduled deployment:

- Start time and date
- End time and date (if necessary)
- Frequency (if necessary)
- Whether or not it is active

You can also display all the scheduled deployments for an OpenDeploy host by selecting **view all** from the **Deployment** list (Figure 44).



Deployment Schedules

Selected Host: localhost Deployment: – view all –

Name	ID	Start Date	Start Time	End Date	End Time	Frequency	Active	Actions
single	1	6/15/01	1:00 PM	10/1/01	1:00 PM	daily	Yes	Edit... Delete... Hold
test	2	6/16/01	12:01 AM			hourly	Yes	Edit... Delete... Hold
reports	0	7/1/01	8:00 AM	12/31/01	11:59 PM	monthly	Yes	Edit... Delete... Hold

New Schedule...

Figure 44: Deployment Schedules Window Displaying All Scheduled Deployments

Viewing Scheduled Deployment Information

To view a deployment schedule, follow these steps:

1. Select **Schedules > View Schedule** to display the Deployment Schedules window.

2. Select the name of the OpenDeploy server whose deployment scheduling information you want to view from the **Selected Host** list.

3. Select the deployment whose scheduling information you want to view from the **Deployment** list, or select **view all** to display all of them.

   The following information is displayed regarding each deployment listed:

   - **Name** — displays the name of the deployment.
   - **ID** — displays the identification number of the scheduled deployment.
   - **Start Date** — displays the day, month, and year specified as the start date when the schedule was added. This may not be the same as the date when the first scheduled deployment will occur.

- **Start Time** — displays the time on the start date specified as the start time when the schedule was added. This may not be the same as the time when the first scheduled deployment will occur.
- **End Date** — displays the day, month, and year specified as the end date when the schedule was added. This may not be the same as the date when the last scheduled deployment will occur.
- **End Time** — displays the time on the end date specified as the end time when the schedule was added. This may not be the same as the time when the last schedule deployment will occur.
- **Frequency** — displays how often the recurring scheduled deployment runs: sub-hourly, hourly, daily, weekly, or monthly.
- **Active** — displays whether or not the scheduled deployment is active.

Editing Scheduled Deployments

To edit a scheduled deployment, follow these steps:

1. Select **Schedules > View Schedule** to display the Deployment Schedules window.
2. Select the name of the OpenDeploy server whose deployment scheduling information you want to view from the **Selected Host** list.
3. Select the deployment whose scheduling information you want to edit from the **Deployment** list. That scheduled deployment is displayed.

   You can also select **view all** to display all the scheduled deployment for the OpenDeploy host.
4. Click **Edit** to display the Edit Schedule window if you want to change any aspect of the existing schedule. The Edit Schedule window looks and functions similarly to the New Schedule window. Here you can change any item of the scheduled deployment.
5. Make you changes and click **Save**.

Deleting Scheduled Deployments

To delete a scheduled deployment, follow these steps:

1. Select **Schedules > View Schedule** to display the Deployment Schedules window.
2. Select the name of the OpenDeploy server whose deployment scheduling information you want to view from the **Selected Host** list.
3. Select the deployment whose scheduling information you want to edit from the **Deployment** list. That scheduled deployment is displayed.

   You can also select **view all** to display all the scheduled deployment for the OpenDeploy host.
4. Click **Delete** to remove the schedule from the scheduler database. You will be prompted to confirm that you want to delete the schedule. If you confirm the deletion, that schedule will be removed from the Deployment Schedules window.

Activating and Deactivating Scheduled Deployments

When you creating a new schedule, it is automatically activated and will run at it scheduled start date. You can stop the scheduled deployment from occurring without deleting it by deactivating it.

To deactivate a scheduled deployment, follow these steps:

1. Select **Schedules > View Schedule** to display the Deployment Schedules window.
2. Select the name of the OpenDeploy server whose deployment scheduling information you want to view from the **Selected Host** list.
3. Select the deployment whose scheduling information you want to edit from the **Deployment** list. That scheduled deployment is displayed.

   You can also select **view all** to display all the scheduled deployment for the OpenDeploy host.
4. Click **Hold** to deactivate that deployment. The **Active** column will display **no** for that scheduled deployment, and the **Hold** button will change to **Activate**.

To reactivate a deactivated scheduled deployment, repeat the same steps you did to deactivate the deployment, and click **Activate**. The **Active** column will display **yes** for that scheduled deployment, and the **Activate** button will change to **Hold**.

Scheduling from the Command Line

You can use OpenDeploy command-line tools to perform the following scheduling-related tasks:

- Add schedules to deployment configurations.
- Delete existing schedules from deployment configurations.
- Display scheduling information on a selected deployment.
- Activate or deactivate a scheduled deployment.

Scheduling command-line tools only can be issued on the host where the OpenDeploy base server software is installed. These commands can be issued by anyone regardless of whether than hold an Administrator or User role. There are no authentication or authorization checks on individuals issuing these commands.

Adding a Schedule

To add a schedule to a deployment using the command line, follow these steps:

1. Navigate to the following directory:

   ```
   od-home/bin
   ```

2. Add a schedule for a deployment by entering the following command at the prompt:

   ```
   iwodschedadd deployment options
   ```

   where *deployment* is the name of the deployment you are scheduling.

There are various options associated with the `iwodschedadd` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodschedadd -h | -v

iwodschedadd deployment [-r [n][m|h|d|w]] [-s [n][m|h|d|w]]
[-e [n][m|h|d|w]]] [-c comment] [-inst instance] [-k "key=value"]+
```

`-h` Displays help information.

`-v` Displays version information.

`deployment` Name of the deployment being scheduled.

`-r` Repeat every *`N`* minutes, hours, days, or weeks.

`-s [`*`N`*`][m|h|d|w]` Time from current time to use as start date. The default is 1 minute from current time when the command is entered.

`-e [`*`N`*`][m|h|d|w]` Amount of time from current time to use as end date. The default end time is none. The scheduled deployment will continue indefinitely.

`n` A numerical value.

`m` Minutes.

`h` Hours.

`d` Days.

`w` Weeks.

`-c` *`comment`* Description of the deployment being scheduled. See "Use of Comments" on page 234 for more information.

`-inst` *`instance`* Includes the deployment instance name *`instance`*, which is a suffix that is appended to the deployment name. This option is used to create unique deployment names for each instance of a deployment configuration. See "Scheduling Deployment Instances" on page 235.

`-k` *`key=value`*	Key/value substitution with "*`key=value`*" as the *`arg`* value. See “Applying Parameter Substitution to Scheduled Deployments” on page 234 for more information.

One-Time Only Deployments

If you only want to run the scheduled deployment once, you do not need to include any option to denote recurrence. In the following example, if you want to schedule the deployment *reports* to deploy a single time a week from now at the same time of day it is currently, you would enter the following command at the prompt:

```
iwodschedadd reports -s 1w
```

Recurring Deployments

If you want your scheduled deployment to run indefinitely at the interval and time you specified, add the `-r` option and the time interval.You can also use the `-s` option and a time period to designate the time of day the deployment will start. Otherwise, the deployment will start at one minute past the time you enter the command. In the following example, if you wanted to schedule the deployment *reports* to run once a day starting at a time one hour from the time you are adding the schedule, you would enter the following command at the prompt:

```
iwodschedadd reports -r 1d -s 1h
```

Recurring Deployments with End Dates

You can specify an end date on which a recurring deployment will cease by including the `-e` option and the amount of time from now that the recurring deployment will cease. If you do not include an end date, the scheduled deployment will occur indefinitely. In the following example, if you wanted the recurring scheduled deployment from the previous example to cease in two weeks, you would enter the following command at the prompt:

```
iwodschedadd reports -r 1d -s 1h -e 2w
```

Use of Comments

You can add a comment to your scheduled deployment using the `-c` option. Your comment can be of any length and include spaces. However, if your comment includes spaces, you must enclose your comment in quotes. In the following example, a comment is added to the previous command:

```
iwodschedadd reports -r 1d -s 1h -e 2w -c "quarterly business report"
```

Comments you add to a scheduled deployment are displayed with its corresponding scheduled deployment when you view deployments using the `iwodschedget` command. This feature is also equivalent to the **Description** box contained in the New Schedule and Edit Schedule windows in the OpenDeploy browser-based user interface.

Applying Parameter Substitution to Scheduled Deployments

You can schedule deployments using parameter substitution, including specifying the parameter values, using `iwodschedadd`. The `iwodschedadd` command supports the `-k` *`parameter=value`* option for parameter substitution in the same manner as `iwodstart`. When you schedule a deployment that uses parameter substitution, you specify the attribute parameter and the substituted value using the following syntax:

```
iwodschedadd deployment ... -k parameter=value
```

In the following example, the deployment *reports* has its `sourceFilesystem` element's area attribute configured in the following manner:

```
sourceFilesystem area="$srcarea^"
```

If you wanted to schedule the deployment to run a single time a week from now at the same time of day it is currently, and also apply the value `C:\temp` to the parameter `srcarea` you would enter the following command at the prompt:

```
iwodschedadd reports -s 1w -k srcarea=C:\temp
```

If either the parameter or its assigned value contained a space, then the entire combined parameter and value must be placed inside of quotation marks. For example, if the value of `srcarea` is:

```
C:\Program Files\monthly
```

then you would enter:

```
iwodschedadd reports -s 1w -k "srcarea=C:\Program Files\monthly"
```

See "Parameter Substitution" on page 198 for a complete description of the parameter substitution feature.

Scheduling Deployment Instances

You can schedule a particular instance of a deployment using the `-inst` *`instance`* option with `iwodschedadd`. Scheduling a deployment instance in this manner uses the following syntax:

```
iwodschedadd deployment -inst instance
```

When you schedule a deployment using the instance feature, the instance name is combined with the deployment name. That combined name is used to track the deployment in the browser-based user interface

See "Specifying a Deployment Instance" on page 103 for a description and usage of the deployment instance feature.

Viewing Scheduled Deployment Information

You can access information on any schedule assigned to your deployment, or all the schedules together, using the `iwodschedget` command-line tool. Several other scheduling-related command-line tools require the schedule ID and other scheduling information that you can get using this tool.

To view information on your deployments schedules, follow these steps:

1. Navigate to the following directory:

```
od-home/bin
```

2. Display the schedule information of a deployment by entering the following command at the prompt:

   ```
   iwodschedget deployment options
   ```

   where *deployment* is the name of the deployment.

There are various options associated with the `iwodschedget` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodschedget -h | -v

iwodschedget -a

iwodschedget -d deployment

iwodschedget -o deployment -j ID
```

`-h` Displays usage information.

`-v` Displays usage information.

`-a` Gets all schedules.This is the default option.

`-d` *`deployment`* Gets all schedules for a particular deployment.

`-o` *`deployment`* Gets one schedule. Requires the deployment name and the deployment ID number.

`deployment` The name of the deployment configuration.

`-j` *`ID`* Specifies a job.The ID number of the deployment. Each time a deployment runs, that deployment is given a unique ID number. Similarly, when you schedule a deployment, that scheduled deployment is also given a issued a unique ID number. Use the `-a` option to see all the ID number for your deployment.

If you wanted to view the schedule information for all of the scheduled deployments residing on your OpenDeploy host, you would enter the following command at the prompt:

```
iwodschedget -a
```

If you wanted to view all schedules for the deployment *reports*, you would enter the following command at the prompt:

```
iwodschedget -d reports
```

If you wanted to view schedule information for the deployment *reports* with an ID number of "2," you would enter the following command at the prompt:

```
iwodschedget -o reports 2
```

Deleting a Schedule

To delete a schedule using the command line, follow these steps:

1. Navigate to the following directory:

   od-home/bin

2. Delete a schedule from a deployment by entering the following command at the prompt:

   iwodscheddelete *deployment options*

   where *deployment* is the name of the deployment.

There are various options associated with the iwodscheddelete command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodscheddelete -h | -v

iwodscheddelete deployment -j ID

iwodscheddelete "dep_name_pattern*" [-j ID]
```

-h	Displays usage information.
-v	Displays version information.
deployment	The name of the deployment configuration.

`-j` *`ID`*

Specifies a job. The ID number of the deployment. Each time a deployment runs, that deployment is given a unique ID number. Similarly, when you schedule a deployment, that scheduled deployment is also given a issued a unique ID number. Use the `iwodschedget -a` command to see all the ID number for your deployment.

`"`*`dep_name_pattern*`*`"`

Deletes schedules based on a wild card name selection, with an optional job identifying number (`-j` option). The wild card pattern must be quoted (`"`*`sample*`*`"`). If the optional job identifying number (`-j` option) is not present, all scheduled deployments beginning with `"`*`dep_name_pattern*`*`"` will be deleted. If the job identifying number is present, only a scheduled deployment beginning with *`dep_name_pattern`* and having a job identifying number equal to the specified value will be deleted.

Because a deployment can have multiple schedules assigned to it, each individual schedule is issued its own unique ID number by OpenDeploy at the time of its creation. You must specify this ID number when you use the `iwodscheddelete` command to ensure that only the schedule you want is being deleted. You can determine this ID value by using the `iwodschedget` command-line tool. See "Viewing Scheduled Deployment Information" on page 235 for more information.

For example, if you wanted to delete a schedule for the deployment *reports* with the ID of "5," you would enter the following command at the prompt:

```
iwodscheddelete reports 5
```

Activating and Deactivating a Schedule

When you creating a new schedule, it is automatically activated and will run at it scheduled start date. You can stop the scheduled deployment from occurring without deleting it by deactivating it.

To activate or deactivate a schedule using the command line, follow these steps:

1. Navigate to the following directory:

   ```
   od-home/bin
   ```

2. Activate or deactivate a scheduled deployment by entering the following command at the prompt:

   ```
   iwodschedactivate deployment options
   ```

   where *deployment* is the name of the deployment.

There are various options associated with the `iwodschedactivate` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodschedactivate -h | -v

iwodschedactivate -a deployment -j ID

iwodschedactivate -a "dep_name_pattern" [-j ID]

iwodschedactivate -d deployment -j ID

iwodschedactivate -d "dep_name_pattern" [-j ID]
```

`-h`	Displays usage information.
`-v`	Displays version information.
`-a` *`deployment`*	Activates a specific scheduled deployment.
`-a "`*`dep_name_pattern*`*`"`	Activates a scheduled deployment with an optional *`jobID`* (`-j` option) using a wild card pattern format. The wild card pattern must be quoted (`"`*`sample*`*`"`). If no `-j` option is present, all scheduled deployments beginning with *`dep_name_pattern`* will be changed.

	If a `-j` option is present, only a scheduled deployment beginning with *`dep_name_pattern`* and having a *`jobID`* equal to the job identifying number will be changed.
`-d` *`deployment`*	Deactivates a specific scheduled deployment, using the *`deployment`* and `-j` *`ID`* options.
`-d "`*`dep_name_pattern*`*`"`	Deactivates a scheduled deployment with an optional job identifying number (`-j` option), using a wild card format. The selection rules are the same as those stated in the schedule activation description above.
`deployment`	The name of the deployment configuration.
`-j` *`ID`*	Specifies a job. The ID number of the deployment. Each time a deployment runs, that deployment is given a unique ID number. Similarly, when you schedule a deployment, that scheduled deployment is also given a issued a unique ID number. Use the `iwodschedget -a` command to see all the ID number for your deployment.

If you wanted to deactivate the scheduled deployment reports with an ID of "5", you would enter the following command at the prompt:

```
iwodschedactivate -d reports 5
```

Conversely, to reactivate the deployment, you would enter the following command at the prompt:

```
iwodschedactivate -a reports 5
```

Reactivating Schedules During or Past Their Effective Period

If a scheduled deployment is deactivated, either by selecting the **Hold** feature in the browser-based user interface, or by running the `iwodschedactivate` command-line tool, reactivating it during or after the effective schedule period results in the following:

- Reactivation during effective period — After the schedule is reactivated, all scheduled runnings of the deployment that have already past are ignored. OpenDeploy will run the next scheduled occurrence of the deployment.
- Reactivation after effective period — OpenDeploy automatically deletes the scheduled deployment without running it.

See "Activating and Deactivating Scheduled Deployments" on page 230 for more information about activating and deactivating scheduled deployments using the browser-based user interface.

See "Activating and Deactivating a Schedule" on page 239 for more information on using the `iwodschedactivate` command-line tool.

Chapter 6

Logging

OpenDeploy provides a variety of different types of logging information including:

- Activities involving the base server or receiver host (base server or receiver log)
- Activities involving a deployment as a whole (macro deployment log) from both the sending and receiving hosts
- Activities involving a specific source/target pair within a deployment (micro deployment log) from both the sending and receiving hosts

You can view and analyze logging information to determine the efficiency of your deployments, whether they are successful or not, and for general troubleshooting.

Log File Location

The default location for all log files is:

```
od-home/log
```

You can specify another location for the base server and receiver host log files by entering the path in the `directory` attribute of the `logRules` element in the corresponding base server or receiver configuration file (by default `odbase.xml` or `odrcvr.xml`). However, you cannot specify a different log file directory location in a deployment configuration. See “Logging Configuration Settings” on page 258 for more information.

Viewing Log Information

You can view log files using one of the following methods:

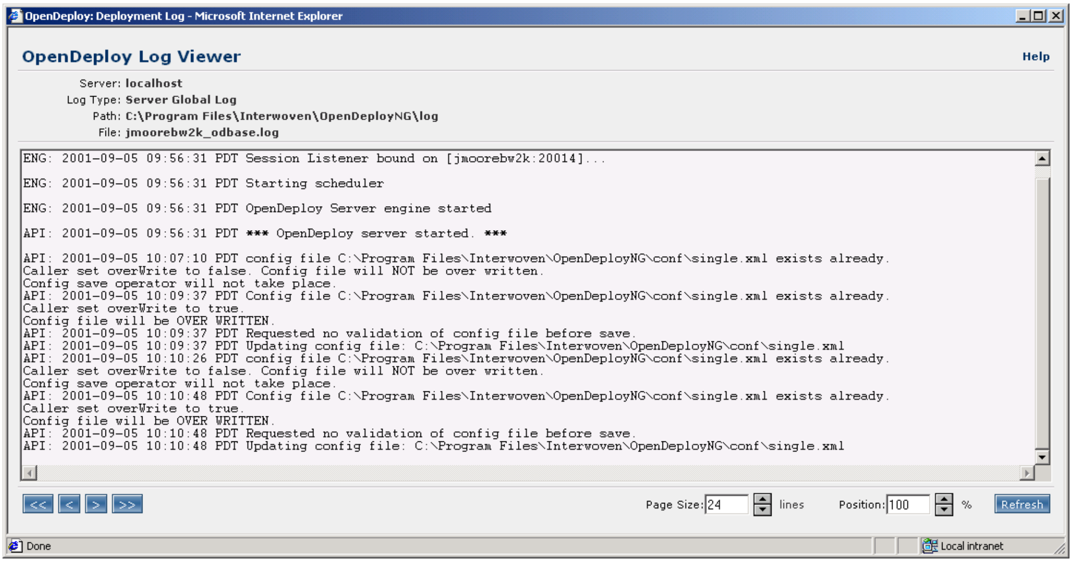
- Text editor
- OpenDeploy user interface

Viewing Log Files from a Text Editor

OpenDeploy log files are standard text files that can be opened with any standard text editor, including vi and Notepad.

Viewing Log Files from the OpenDeploy User Interface

You can view log files in the OpenDeploy user interface using the OpenDeploy Log Viewer window (Figure 45). The OpenDeploy Log Viewer window is a separate browser window that appears when you click a **View Log** button in a window.



Server: localhost
Log Type: Server Global Log
Path: C:\Program Files\Interwoven\OpenDeployNG\log
File: jmoorebw2k_odbase.log

```
ENG: 2001-09-05 09:56:31 PDT Session Listener bound on [jmoorebw2k:20014]...

ENG: 2001-09-05 09:56:31 PDT Starting scheduler

ENG: 2001-09-05 09:56:31 PDT OpenDeploy Server engine started

API: 2001-09-05 09:56:31 PDT *** OpenDeploy server started. ***

API: 2001-09-05 10:07:10 PDT config file C:\Program Files\Interwoven\OpenDeployNG\conf\single.xml exists already.
Caller set overWrite to false. Config file will NOT be over written.
Config save operator will not take place.
API: 2001-09-05 10:09:37 PDT Config file C:\Program Files\Interwoven\OpenDeployNG\conf\single.xml exists already.
Caller set overWrite to true.
Config file will be OVER WRITTEN.
API: 2001-09-05 10:09:37 PDT Requested no validation of config file before save.
API: 2001-09-05 10:09:37 PDT Updating config file: C:\Program Files\Interwoven\OpenDeployNG\conf\single.xml
API: 2001-09-05 10:10:26 PDT config file C:\Program Files\Interwoven\OpenDeployNG\conf\single.xml exists already.
Caller set overWrite to false. Config file will NOT be over written.
Config save operator will not take place.
API: 2001-09-05 10:10:48 PDT Config file C:\Program Files\Interwoven\OpenDeployNG\conf\single.xml exists already.
Caller set overWrite to true.
Config file will be OVER WRITTEN.
API: 2001-09-05 10:10:48 PDT Requested no validation of config file before save.
API: 2001-09-05 10:10:48 PDT Updating config file: C:\Program Files\Interwoven\OpenDeployNG\conf\single.xml
```

Page Size: 24 lines   Position: 100 %

Figure 45: OpenDeploy Log Viewer Window

Each log you select to view is displayed in a separate browser window which allows you to view multiple logs simultaneously.

The format and structure of the various logs are essentially the same. The deployment log windows include the name of the deployment associated with the logs. Here is a description of the log windows:

- **Server** — displays the name of the OpenDeploy host sending and receiving the deployment.
- **Log Type** — displays the type of log file being displayed, such as a server global log (base server or receiver host log) or a deployment macro or micro log for a deployment sent or received.
- **Deployment** (deployment logs only) — displays the name of the deployment associated with the displayed log.
- **Path** — displays the absolute path to the directory containing the log file being displayed.
- **File** — displays the name of the log file being displayed.The following types of log files can be displayed in this window:
  - `server_odbase.log` — indicates the log file is a base server log.
  - `server_odrcvr.log` — indicates the log file is a receiver log.
  - `src.deployment.log` — indicates the log file is a source macro deployment log.
  - `rcv.deployment.definition.target-server.log` — indicates the log file is a receiver macro deployment log.
  - `src.deployment.definition.source-server.to.target-server.log` — indicates the log file is a source micro deployment log.
  - `rcv.deployment.definition.source-server.to.target-server.log` — indicates the log file is a receiver micro deployment log.

  where the following variables apply:
  - *`deployment`* — the name of the associated deployment.
  - *`definition`* — the name of the definition in the deployment configuration that contains the source/target pairing.
  - `source-server` — the name of the source host sending the deployment.
  - *`target-server`* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.

- << button — click to display the beginning of the log.
- < button — click to display the previous portion of the log.
- > button — click to display the next portion of the log.
- >> button — click to display the end of the log.
- **Page Size** box — enter the number of lines of the deployment log you want to view. You can enter the exact number, or click the arrow buttons up and down in increments of 10 from the existing number. You can range in size from 10 to 1000 lines. You must click **Refresh** to implement the number you entered.
- **Position** box — enter the proportional location percentage (0-100) of the log file to be displayed. You can enter the exact number, or click the arrow buttons up and down in increments of 10. For example, the beginning of the log would be `0`, while the center would be `50`. You must click **Refresh** to implement the number you entered.
- **Refresh** button — click to refresh the log and to read in fresh data with the **Page Size** and **Position** values you entered.

Base Server Logging

All activities concerning the OpenDeploy base server host are written to the *base server log*. Base server log entries include information on:

- Starting up OpenDeploy services and daemons
- Adding, removing, and modifying the Administrator and User roles of individuals
- Starting deployments
- Receiving deployments
- Adding schedules for deployments
- Starting a scheduled deployment

- Requests from individuals with User roles that have been denied due to insufficient authorization
- Error information on requested operations

Reviewing the base server log is an effective method of determining the activities of your OpenDeploy sending host, and of troubleshooting problems.

Here is an example of base server log entries:

```
BEGIN LOG: Logfile [C:\Interwoven\OpenDeployNG\log\jmoorebw2k_odbase.log] -------
--
API: 2001-11-12 13:09:55 PST GMT-08:00 Using time zone: Pacific Standard Time
API: 2001-11-12 13:09:55 PST GMT-08:00 Using locale: en_US
API: 2001-11-12 13:09:55 PST GMT-08:00 Using OpenDeploy home directory:
C:\INTERW~1\OPENDE~1
API: 2001-11-12 13:09:55 PST GMT-08:00 Using server config file specified in
deploy.cfg: C:\INTERW~1\OPENDE~1\etc\odbase.xml
API: 2001-11-12 13:09:55 PST GMT-08:00 Using server nodes config file specified in
deploy.cfg: C:\INTERW~1\OPENDE~1\etc\odnodes.xml
API: 2001-11-12 13:09:59 PST GMT-08:00 Using server log directory
C:\Interwoven\OpenDeployNG\log specified in server config file.
API: 2001-11-12 13:09:59 PST GMT-08:00 Using OpenDeploy Server log file
C:\Interwoven\OpenDeployNG\log\jmoorebw2k_odbase.log.
```

By default, the base server log file resides in the following location:

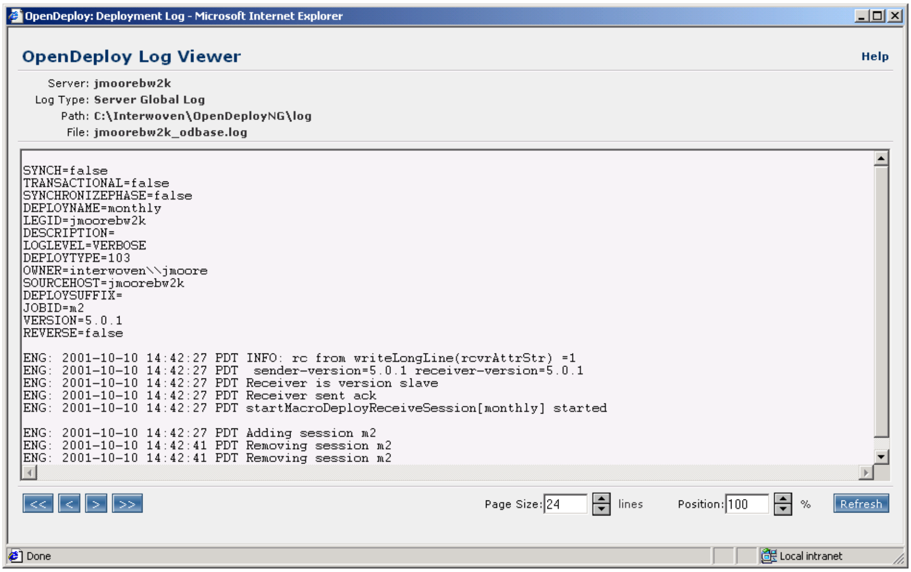
`od-home/log/server_odbase.log`

where *server* is the name of the host server. If your OpenDeploy host was named *mars*, then the base server log file path and name would be:

`od-home/log/mars_odbase.log`

To access the base server log from the user interface, follow these steps:

1. Select **Servers > Manage Servers** to display the Manage Servers window.
2. Select the host whose source base server log you want to view from the **Selected Host** list.
3. Click **View Log**. A separate browser window appears displaying the OpenDeploy Log Viewer window containing the base server log (Figure 47).



Server: jmoorebw2k
Log Type: Server Global Log
Path: C:\Interwoven\OpenDeployNG\log
File: jmoorebw2k_odbase.log

```
SYNCH=false
TRANSACTIONAL=false
SYNCHRONIZEPHASE=false
DEPLOYNAME=monthly
LEGID=jmoorebw2k
DESCRIPTION=
LOGLEVEL=VERBOSE
DEPLOYTYPE=103
OWNER=interwoven\\jmoore
SOURCEHOST=jmoorebw2k
DEPLOYSUFFIX=
JOBID=m2
VERSION=5.0.1
REVERSE=false

ENG: 2001-10-10 14:42:27 PDT INFO: rc from writeLongLine(rcvrAttrStr) =1
ENG: 2001-10-10 14:42:27 PDT  sender-version=5.0.1 receiver-version=5.0.1
ENG: 2001-10-10 14:42:27 PDT Receiver is version slave
ENG: 2001-10-10 14:42:27 PDT Receiver sent ack
ENG: 2001-10-10 14:42:27 PDT startMacroDeployReceiveSession[monthly] started

ENG: 2001-10-10 14:42:27 PDT Adding session m2
ENG: 2001-10-10 14:42:41 PDT Removing session m2
ENG: 2001-10-10 14:42:41 PDT Removing session m2
```

Page Size: 24 lines Position: 100 %

Figure 46: Base Server Log

Receiver Logging

All activities concerning the OpenDeploy receiver host are written to the *receiver* log. Receiver log entries include information on:

- Starting up OpenDeploy services and daemons
- Receiving deployments

Reviewing the receiver log is an effective method of determining the activities of your OpenDeploy receiving host, and of troubleshooting problems.

By default, the log file resides in the following location:

```
od-home/log/server_odrcvr.log
```

where `server` is the name of the receiver host. If your OpenDeploy host was named *venus*, then the receiver log file path and name would be:

```
od-home/log/venus_odrcvr.log
```

You can view the receiver log from the OpenDeploy user interface in the same manner as for the base server log. See "Base Server Logging" on page 246 for more information.

Macro Deployment Logging

The *macro deployment* logs write entries on every aspect of a deployment each time it is run. There are two macro deployment logs, one for the source host (the source macro deployment log) and one for the target host (the receiver macro deployment log). If the deployment is configured as a fan-out deployment with multiple target hosts, the macro deployment log will have entries written for each source/target host pairing. Each new running of a deployment causes a new set of log entries to be appended onto the file, so you can review the history of the deployment over a period of time.

Macro deployment log entries include information on:

- Whether or not deployments to each target host were successful.
- Time the deployments took.

Reviewing the macro deployment log is a way to determine how a particular deployment functions, and to troubleshoot problems with that deployment. Here is an example of macro deployment log entries:

```
NG: 2001-11-28 13:06:12 PST GMT-08:00
internalDepName=.monthly.MYDEFINITIONNAME.jmoorebw2k
ENG: 2001-11-28 13:06:12 PST GMT-08:00 Got converted config for
.monthly.MYDEFINITIONNAME.jmoorebw2k
ENG: 2001-11-28 13:06:12 PST GMT-08:00 Waiting for 2 children to complete phases
ENG: 2001-11-28 13:06:12 PST GMT-08:00 All 2 children completed their phases
ENG: 2001-11-28 13:06:12 PST GMT-08:00 Deployment[monthly]   Elapsed Time=120 ms
ENG: 2001-11-28 13:06:12 PST GMT-08:00 End logfile
[C:\Interwoven\OpenDeployNG\log\src.monthly.log]
```

Source Macro Deployment Log

The *source macro deployment log* file contains log entries for a deployment where the OpenDeploy host is the sender.

The source macro log by default resides in the following location:

`*od-home*/log/src.*deployment*.log`

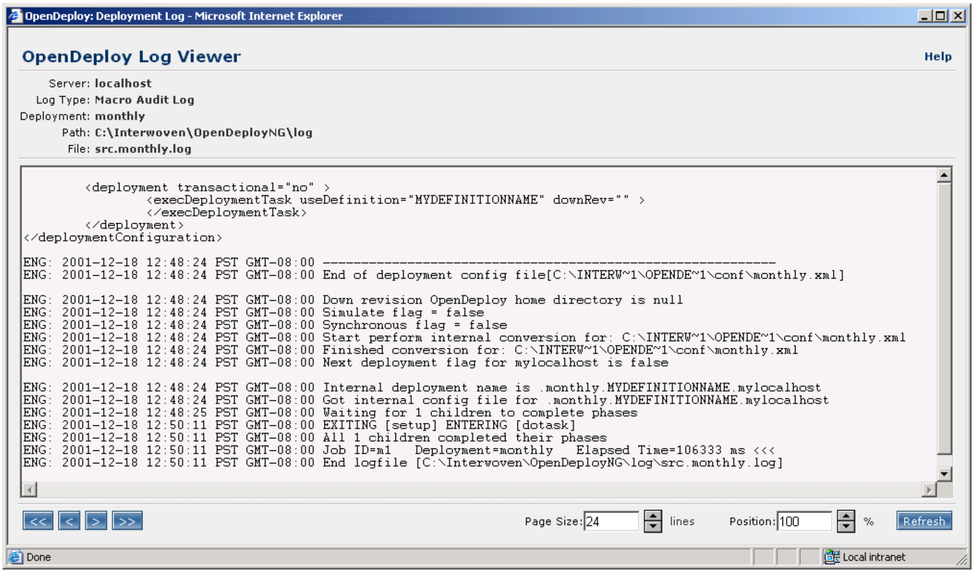
where *deployment* is the name of the deployment configuration. If your deployment was named *monthly*, then the source macro deployment log file path and name would be:

`*od-home*/log/src.monthly.log`

To access the source macro deployment log from the user interface, follow these steps:

1. Select **Deployment > View Deployment** to display the Source Deployment window.
2. Select the host containing the deployment whose source macro deployment log you want to view from the **Selected Host** list.
3. Select **Sending** from the **View** list. All the deployments that the host sends are displayed in a table.

4. Click **View Log** for the deployment whose source macro deployment log you want to view. A separate browser window appears displaying the OpenDeploy Log Viewer window containing the source macro deployment log (Figure 47).



Server: localhost
Log Type: Macro Audit Log
Deployment: monthly
Path: C:\Interwoven\OpenDeployNG\log
File: src.monthly.log

```
        <deployment transactional="no" >
                <execDeploymentTask useDefinition="MYDEFINITIONNAME" downRev="" >
                </execDeploymentTask>
        </deployment>
</deploymentConfiguration>

ENG: 2001-12-18 12:48:24 PST GMT-08:00 ------------------------------------------------------------
ENG: 2001-12-18 12:48:24 PST GMT-08:00 End of deployment config file[C:\INTERW~1\OPENDE~1\conf\monthly.xml]

ENG: 2001-12-18 12:48:24 PST GMT-08:00 Down revision OpenDeploy home directory is null
ENG: 2001-12-18 12:48:24 PST GMT-08:00 Simulate flag = false
ENG: 2001-12-18 12:48:24 PST GMT-08:00 Synchronous flag = false
ENG: 2001-12-18 12:48:24 PST GMT-08:00 Start perform internal conversion for: C:\INTERW~1\OPENDE~1\conf\monthly.xml
ENG: 2001-12-18 12:48:24 PST GMT-08:00 Finished conversion for: C:\INTERW~1\OPENDE~1\conf\monthly.xml
ENG: 2001-12-18 12:48:24 PST GMT-08:00 Next deployment flag for mylocalhost is false

ENG: 2001-12-18 12:48:24 PST GMT-08:00 Internal deployment name is .monthly.MYDEFINITIONNAME.mylocalhost
ENG: 2001-12-18 12:48:24 PST GMT-08:00 Got internal config file for .monthly.MYDEFINITIONNAME.mylocalhost
ENG: 2001-12-18 12:48:25 PST GMT-08:00 Waiting for 1 children to complete phases
ENG: 2001-12-18 12:50:11 PST GMT-08:00 EXITING [setup] ENTERING [dotask]
ENG: 2001-12-18 12:50:11 PST GMT-08:00 All 1 children completed their phases
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Job ID=m1   Deployment=monthly   Elapsed Time=106333 ms <<<
ENG: 2001-12-18 12:50:11 PST GMT-08:00 End logfile [C:\Interwoven\OpenDeployNG\log\src.monthly.log]
```

Page Size: 24 lines   Position: 100 %

Figure 47: Source Macro Deployment Log

Receiver Macro Deployment Log

The *receiver macro deployment log* provides a similar service for OpenDeploy hosts receiving deployments as the source macro deployment log does for sending hosts. The receiver macro deployment log contains macro-type entries for the deployments received by the host.

A separate receiver macro log is generated anytime the combination of deployment name, definition name, and logical target name is unique. For example, if a deployment has three separate definitions all pointed at the same target host, three separate receiver macro log files will be generated on the host receiving the deployment.

The receiver macro log by default resides in the following location:

```
od-home/log/rcv.deployment.definition.target-server.log
```

where the following variables apply:

- `deployment` — the name of the associated deployment.
- `definition` — the name of the definition in the deployment configuration that contains the source/target pairing.
- `target-server` — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.

If your deployment was named *monthly* and the definition was named *corporate*, and the target host's logical name is *jupiter*, then the receiver macro deployment log file path and name would be:

```
od-home/log/rcv.monthly.corporate.jupiter.log
```

You must select **Receiving** from the **View** list in the Source Deployments window to access receiver macro deployment logs. See "Source Macro Deployment Log" on page 250 for more information.

Micro Deployment Logging

The *micro deployment* logs write entries for each source/target host pair in a deployment. If the deployment includes only a single source host and target host, then one micro deployment log each is generated on the source and target hosts. If the deployment is a fan-out type with several target hosts, then a micro deployment log is generated for each of those target hosts.

The source host will generate a separate micro deployment log (the source micro deployment log) for each target host. Each target host receiving the deployment generates its own log (the receiver micro deployment log). Each running of the deployment results in a new set of log entries to be appended onto the file, so you can review the history of the deployment over multiple runnings.

Micro deployment log entries include information on:

- Contact made with the source or target host
- Directories and files successfully deployed
- Directories and files that failed to deploy

Reviewing the micro deployment log is a way to determine how a particular deployment functions, and to troubleshoot problems with that source or target host participating in a deployment. Here is an example of macro deployment log entries:

```
Directories deployed : 2  Files deployed : 34 Links deployed : 0
Directories failed   : 0  Files failed   : 0  Links failed   : 0
Directories deleted  : 0  Files deleted  : 0  Links deleted  : 0
id=0 server: File Content transferred: 4647780 bytes
id=0 server: [Wed Jun 13 10:29:55 2001] Deployment COMPLETED
```

Source Micro Deployment Log

The *source micro deployment log* contains log entries for the movement of files between the source host and one target host. If the deployment is a fan-out deploying to several target hosts, each source/target deployment will log its own micro deployment log.

The source micro log by default resides in the following location:

```
od-home/log/src.deployment.definition.source-server.to.target-server.log
```

where the following variables apply:

- *deployment* — the name of the associated deployment.
- *definition* — the name of the definition in the deployment configuration that contains the source/target pairing.
- *source-server* — the name of the source host sending the deployment.
- *target-server* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.

If your deployment was named *monthly*, the definition named *corporate*, your sending host named *mars*, and the target host named *venus*, then the source micro deployment log file name would be:

```
src.monthly.corporate.mars.to.venus.log
```

If your fan-out deployment had following targets:

- *venus*
- *jupiter*

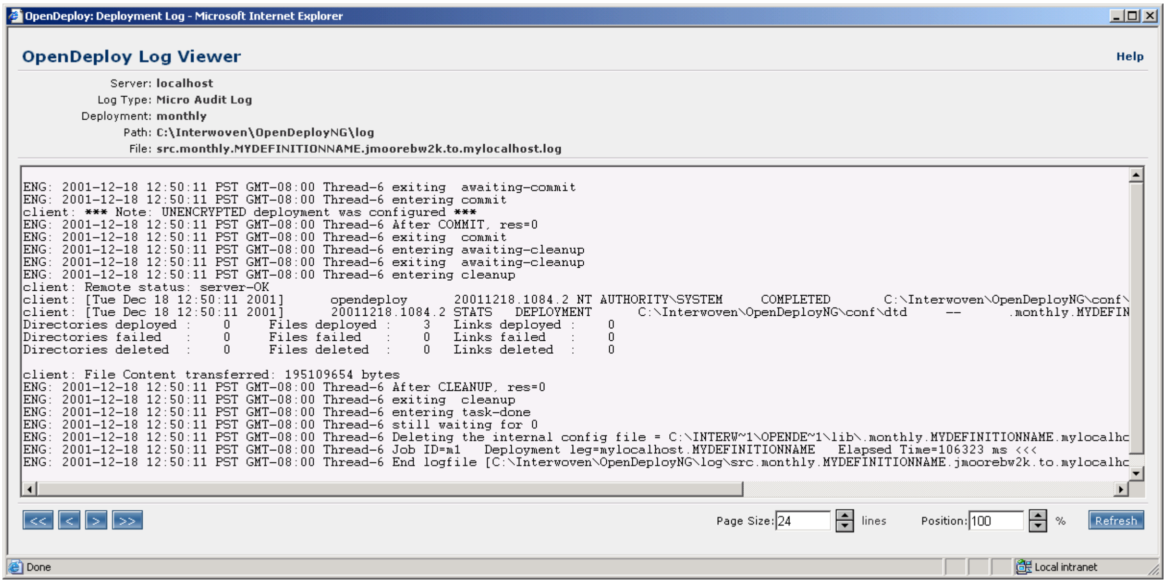
then the sending host would have the two corresponding source micro deployment log files:

```
src.monthly.corporate.mars.to.venus.log

src.monthly.corporate.mars.to.jupiter.log
```

To access the source micro deployment log from the user interface, follow these steps:

1. Select **Deployment > View Deployment** to display the Source Deployment window.
2. Select the host containing the deployment whose source macro deployment log you want to view from the **Selected Host** list.
3. Select **Sending** from the **View** list. All the deployments that the host sends are displayed in a table.
4. Click the link of the deployment whose source micro deployment log you want to view. The **Details** table appears at the bottom of the window, displaying a separate row for each target host of the deployment.
5. Click **View Log** for the appropriate target host. A separate browser window appears displaying the OpenDeploy Log Viewer window containing the source micro deployment log (Figure 48).



Server: localhost
Log Type: Micro Audit Log
Deployment: monthly
Path: C:\Interwoven\OpenDeployNG\log
File: src.monthly.MYDEFINITIONNAME.jmoorebw2k.to.mylocalhost.log

```
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 exiting  awaiting-commit
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 entering commit
client: *** Note: UNENCRYPTED deployment was configured ***
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 After COMMIT, res=0
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 exiting  commit
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 entering awaiting-cleanup
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 exiting  awaiting-cleanup
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 entering cleanup
client: Remote status: server-OK
client: [Tue Dec 18 12:50:11 2001]      opendeploy      20011218.1084.2 NT AUTHORITY\SYSTEM     COMPLETED       C:\Interwoven\OpenDeployNG\conf\
client: [Tue Dec 18 12:50:11 2001]      20011218.1084.2 STATS   DEPLOYMENT      C:\Interwoven\OpenDeployNG\conf\dtd     --      .monthly.MYDEFIN
Directories deployed :    0     Files deployed :    3   Links deployed :    0
Directories failed   :    0     Files failed   :    0   Links failed   :    0
Directories deleted  :    0     Files deleted  :    0   Links deleted  :    0

client: File Content transferred: 195109654 bytes
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 After CLEANUP, res=0
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 exiting  cleanup
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 entering task-done
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 still waiting for 0
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 Deleting the internal config file = C:\INTERW~1\OPENDE~1\lib\.monthly.MYDEFINITIONNAME.mylocalhc
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 Job ID=m1   Deployment leg=mylocalhost.MYDEFINITIONNAME   Elapsed Time=106323 ms <<<
ENG: 2001-12-18 12:50:11 PST GMT-08:00 Thread-6 End logfile [C:\Interwoven\OpenDeployNG\log\src.monthly.MYDEFINITIONNAME.jmoorebw2k.to.mylocalhc
```

Page Size: 24 lines   Position: 100 %

Figure 48: Source Micro Deployment Log

Receiver Micro Deployment Log

The *receiver micro deployment log* provides a similar service for OpenDeploy hosts receiving deployments as the source micro deployment log does for sending hosts. The receiver micro deployment log contains entries regarding the movement of files between the source and target hosts during the deployment.

The receiver micro log by default resides in the following location:

```
od-home/log/rcv.deployment.definition.source-server.to.target-server.log
```

where the following variables apply:

- *deployment* — the name of the associated deployment.
- *definition* — the name of the definition in the deployment configuration that contains the source/target pairing.
- *source-server* — the name of the source host sending the deployment.
- *target-server* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.

If your deployment was named *monthly*, the definition named *corporate*, your sending host named *mars*, and the target host named *venus*, then the receiver micro deployment log file name would be:

```
rcv.monthly.corporate.mars.to.venus.log
```

You must select **Receiving** from the **View** list in the Source Deployments window to access micro deployment logs. See “Source Micro Deployment Log” on page 254 for more information.

Logging Levels

OpenDeploy provides the following logging level options:

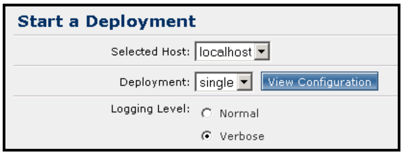
- Verbose — logs high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
- Normal — logs standard status and error messages. In most cases, this level of logging provides a sufficient amount of detail to meet your needs.

You can configure logging settings both on an OpenDeploy server basis, and on a deployment configuration basis. See "Logging Configuration Settings" on page 258 for more information.

Settings for deployment logging in the base server or receiver configuration can be overridden in the user interface, or by the deployment configuration. See "Logging Rules Hierarchy" on page 260 for more information.

Defining Logging Levels in the User Interface

Any time you manually start a deployment from the OpenDeploy user interface (Figure 49), you can specify the level of logging for that deployment. A level specified here automatically overrides any logging levels specified in the base server or deployment configurations.



Start a Deployment

Selected Host: localhost

Deployment: single

Logging Level: ○ Normal ● Verbose

Figure 49: Log Levels in the User Interface

Defining Logging Levels from the Command Line

You can specify the logging level for a deployment you start using the `iwodstart` command-line tool by including the `-V` option and the desired logging level. For example:

```
iwodstart deployment -V verbose or

iwodstart deployment -V normal
```

See "Running Deployments from the User Interface" on page 87 for more information on using `iwodstart` to run deployments.

Logging Configuration Settings

You can configure logging in both base server and receiver host configuration files (by default `odbase.xml` and `odrcvr.xml`) and in individual deployment configurations (for example, `test.xml`). Defining the logging settings in the configurations automates the logging rules that apply when a deployment runs. Logging settings are defined in the `logRules` element and its associated attributes.

Base Server and Receiver Configurations

In the base server and receiver configuration file, the `logRules` element appears as:

```
<logRules maxBytes="x" level="y" directory="z" />
```

where *x*, *y*, and *z* are the values for the following attributes:

- `maxBytes` — specifies the maximum size in bytes a log file is allowed to grow before the file is closed and OpenDeploy begins writing to a new file. This value is known as the *rollover threshold*. The default `maxBytes` value is 32 megabytes. You can specify different byte measurements in the value, including megabytes (`mb`), kilobytes (`kb`), and bytes (`b`). For example:

  ```
  maxBytes="10mb" or

  maxBytes="10000kb" or

  maxBytes="10000000b"
  ```

indicates that the log file size can grow to 10 megabytes before OpenDeploy will close that log file and start a new one.

Ensure that you include the proper measurement indicator when setting the threshold size. If no recognizable size measurement is indicated, OpenDeploy uses its default value instead. For example, if the following value was specified:

```
maxBytes="10"
```

OpenDeploy would ignore that stated value and use the default value (`32mb`) instead.

If the unit of measure is present but unrecognized by OpenDeploy, the default value is used. For example, if the following value was specified:

```
maxBytes="1000x"
```

OpenDeploy would ignore this value and use the default value (`32mb`).

OpenDeploy will not honor a `maxBytes` value of less than 100 kilobytes (`100kb`). For example, if the following value was specified:

```
maxBytes="50kb"
```

OpenDeploy would ignore this value and use the default value (`32mb`) instead.

See “Log File Size Management” on page 261 for more information on rollover threshold.

- `level` – indicates the level and type of logging OpenDeploy will perform.
  - `verbose` — logs high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
  - `normal` — logs standard status and error messages. In most cases, this level of logging provides a sufficient amount of detail to meet your needs.
- `directory` (base server and receiver configuration only) — specifies the absolute path directory location for log files. The default location is:

  `od-home``/log`

Deployment Configurations

The `logRules` element functions the same in a deployment configuration as it does in a base server or receiver configuration file, except that the `directory` attribute is not present. For example:

```
<logRules maxBytes="10mb" level="normal" />
```

You can only specify an alternative log file home in the base server or receiver configuration file. Logging settings for macro and micro deployment logs in a deployment configuration will override logging settings in the base server or receiver configuration.

Logging Rules Hierarchy

The following logging rules hierarchy apply to logging rules:

Base Server and Receiver Host Logging

The logging levels for the base server and receiver logs are specified in their associated configurations. The level of logging is defined as the value of the `level` attribute in the `logRules` element. Logging levels for this type of logging cannot be overridden. If the `logRules` element or any of its attributes are absent from the base server or receiver configuration, OpenDeploy will use the following default settings:

- `level` — `verbose`
- `maxBytes` — `32 mb`
- `directory` — *`od-home`*`/logs`

Macro and Micro Deployment Logging

The following hierarchy applies to the logging verbosity and maximum file size for deployment macro and micro logs:

- Logging levels specified in the OpenDeploy user interface or the `iwodstart` command-line tool take precedence.
- If the previous parameters are not specified, logging settings in the deployment configuration take precedence.
- If neither of the previous parameters are specified, logging settings in the base server and receiver files take precedence.
- If no other parameters are specified, OpenDeploy will use its default logging settings. See "Logging Configuration Settings" on page 258 for more information.
- If there are any syntax errors in the specified maximum bytes value, such as a unit of measurement being absent or unreadable, OpenDeploy will use its default values for these circumstances. See "Logging Configuration Settings" on page 258 for more information.

Log File Size Management

Log files can grow large quickly, especially with large or numerous deployments. Using verbose logging (the default logging level) can also generate large log files.You should determine how much storage space to devote to log files before setting the logging type. OpenDeploy uses a log file *rollover threshold* feature to determine the maximum size a single log file can grow before that file is closed to new log entries and a new log file is generated.

The deployment macro logs for the source host and the target hosts are linked for rolling over. When the source host's macro log file requires being rolled over because it has met or exceeded its rollover threshold, the corresponding deployment macro log on the receiving host will also be rolled over, even if it has not reached its rollover limit. The source host determines when a rollover is required.

The deployment micro logs are rolled over in a manner similar to that of macro logs. The source host determines when a log file rollover must occur, and both the source and target micro logs are rolled over together. If a deployment is a fan-out type that includes multiple source/target pairs, the logs of each source/target pair are rolled over independent of other target-source pairs.

Rollover Threshold Size Determination

The threshold size of the log file is specified in the `logRules` element's `maxBytes` attribute in the base server and receiver configurations files, and in the deployment configurations. If that value is not specified, or if the element is not defined in the configuration, OpenDeploy will look to the same element in the base server configuration file for logging information. If that information is still not present, OpenDeploy will use the default size of 32 MB. See "Logging Configuration Settings" on page 258 for more information.

Rolled Over Log File Naming

OpenDeploy will roll over a log file when it detects the file size exceeds its specified rollover size as indicated by its `maxBytes` attribute value. A serial naming convention is used to indicate the order of the archived log files. OpenDeploy uses a counter file (*`counter`*`.cnt`) to manage the generation of log archive files. Do not move or delete the counter file from the log directory.

When the log file is rolled over, that log's file name is appended with a four-digit extension. This extension starts at `0001` and increases by 1 each time the same log is rolled over. Each log has a separate counter to keep track of rollovers. OpenDeploy subsequently creates a new log file with the original log file name and will start writing log entries to it. For example, if the following log file:

```
src.monthly.log
```

reached its rollover threshold level, OpenDeploy would close this file to further entries and subsequently archive it by adding an appropriate four-digit suffix:

```
src.monthly.log1234
```

In the following example, the log file `src.single.mars.to.venus.log` has been archived four times:

```
4669 Jun 6 10:49 src.single.mars.to.venus.log (current log)
5 Jun 6 10:49 src.single.mars.to.venus.log.cnt (counter)
3877 May 15 15:40 src.single.mars.to.venus.log0001 (first archive)
2126 May 15 15:40 src.single.mars.to.venus.log0002 (second archive)
2126 May 15 15:42 src.single.mars.to.venus.log0003 (third archive)
3901 May 23 13:39 src.single.mars.to.venus.log0004 (fourth archive)
```

When the log rollover extension reaches `9999`, the next time it rolls the log over, it will reset to `0001`, followed by `0002` and so forth. If the log file with suffix `0001` already exists, that file will be overwritten by the new one as the extension resets. If you want to preserve old log files that are at risk of being overwritten, you should move them to another location.

Log File Recovery

The following sections explain log file recovery in OpenDeploy.

Base Server and Receiver Log Files

If the OpenDeploy base server or receiver log file is deleted, OpenDeploy will detect that it is missing and create a new log when one of the following event occurs:

- When you start a deployment manually from the OpenDeploy user interface or using the `iwodstart` command line tool, or if a scheduled deployment is run.
- When you refresh the host through the OpenDeploy user interface of the `iwodserverreset` command-line tool. If the OpenDeploy base server or receiver configuration files have not been changed, this is a convenient way to generate new server log files if the existing ones become lost or damaged.

- When any of the following security related events occur on the OpenDeploy host:
  - The list of users in a role is viewed.
  - A user is added or removed from a role.
  - A deployment is added or removed from an user in the `od-user` role.
  - A user is denied access to an OpenDeploy function.
- When the OpenDeploy host is restarted after having been stopped.

Deployment Log Files

OpenDeploy will automatically generate new deployment macro and micro log files on both the source and receiver hosts any time existing deployment log files are not detected. If a deployment log file is lost or damaged while that deployment is in progress, no recovery is possible. However, because deployments are logged on both the sending and receiving hosts, you can view the remaining logs.

Reporting Logging

There are several log files associated with the OpenDeploy reporting feature. These log files, their locations, and configurations are describeded in Chapter 7, “Reporting” under the following sections:

- Host reporting log — see “Logging” on page 269.
- Reporting management log — see “Logging” on page 276.

Chapter 7

Reporting

Each OpenDeploy base server and receiver installation includes a reporting component used to publish events to a reporting server, which is installed as part of the administration package. Events sent by an OpenDeploy host to the reporting server are stored in a user-defined database. These events can subsequently be accessed by the administration server for viewing within the browser-based user interface.

Reports generated by an OpenDeploy host are durable. If the reporting server is temporarily unavailable, the OpenDeploy host retains the events until they can be successfully transferred after the reporting server goes back into service.

OpenDeploy provides the following reporting features available through the browser-based user interface:

- Custom reports — reports that allow you to apply a search criteria based on deployment name, deployment owner, timeframe, and other factors.
- SQL query reports — reports where you compose the structure of the report yourself using SQL. You can also apply the search criteria feature available in custom reports to your SQL query reports.
- Quick reports — queries of either type that are saved and available for running at any time without having to perform any additional configuration.

Host Configuration

Each OpenDeploy host participating in reporting must have the following:

- The `eventReporting` element must be enabled in the host configuration file.
- The host reporting configuration file must be fully configured for reporting.

Host Configuration File

Each OpenDeploy base server (by default `odbase.xml`) or receiver (by default `odrcvr.xml`) configuration file includes the `eventReporting` element, which enables the host's ability to use the reporting feature and specifies host reporting configuration file:

```
<deployServerConfiguration>
   ...
   <eventReporting
      enabled="yes"
      cfgPath="C:\Interwoven\OpenDeployNG\etc\eventReportingConfig.xml"
      />
</deployServerConfiguration>
```

Refer to Chapter 2, "Host Configuration" on page 47 of the *OpenDeploy Installation and Reference Guide* for more information on host configuration files.

Enabling Reporting

You must enable the reporting feature in the host configuration file by giving the `eventReporting` element's `enabled` attribute a value of `yes`. If the enabled attribute has a value of `no`, or if the `eventReporting` element is missing from the host configuration, reporting is not enabled on that host.

During the base server and receiver software installation, you are prompted whether to enable the reporting feature or not. Typically, reporting is intended to capture events from the original sending host, and perhaps next-tier base servers, but not necessarily end point targets. Therefore, the default installation for the base server software is with reporting enabled, while the default installation for receiver software is with reporting disabled.

Path to Host Reporting Configuration File

The `eventReporting` element's `cfgPath` attribute value specifies the path to the reporting configuration file (by default `eventReportingConfig.xml`). The default path is the following:

```
od-home/etc/eventReportingConfig.xml
```

but you can name and locate the file anywhere on your host's file system, as long as that name and location are reflected in the `cfgPath` attribute.

The `eventReporting` element is included in the base server configuration file by default, automatically enabling the feature. To disable reporting, you must comment out or delete the `eventReporting` element from the base server configuration file.

Host Reporting Configuration File

Reporting is a highly flexible feature that requires its own configuration file on each OpenDeploy host. The host's reporting configuration file (by default `eventReportingConfig.xml`) contains the elements and attributes that determine the functionality of the reporting feature on that host.

The host reporting configuration file contains various elements and attributes that manages the host's ability to use the reporting feature. The following sections describes those elements and attributes you typically would want to configure, such as logging. Other elements and attributes require no user configuration and are not covered. To obtain information on all elements and attributes contained in the host reporting configuration file, refer to Chapter 6, "Reporting Server Configuration DTD" on page 169 of the *OpenDeploy Installation and Reference Guide*.

The following page displays a sample host reporting configuration file.

```
<eventReportingConfiguration>
 <log name="openDeployPublisherLog"
         path="MYOPENDEPLOY/log/MYHOSTNAME_publisher.log"
         append="true"/>

    <process name="OpenJMS"
             startCommand="MYOPENDEPLOY/jre/bin/java -Djava.security.manager -
Djava.security.policy=MYOPENDEPLOY/lib/openjms.policy -
Djava.rmi.server.codebase=file://MYOPENDEPLOY/openjms/lib/openjms-rmi-0.7.2.jar -
Djava.rmi.server.hostname=MYHOSTNAME ${OPENJMS_RMI_PORTS} -classpath
${OPENJMS_CP} org.exolab.jms.server.JmsServer -config MYOPENDEPLOY/etc/
jmsConfig.xml"
             stopCommand="MYOPENDEPLOY/jre/bin/java -Djava.security.manager -
Djava.security.policy=MYOPENDEPLOY/lib/openjms.policy ${OPENJMS_RMI_PORTS} -
classpath ${OPENJMS_CP} org.exolab.jms.administration.AdminMgr -config
MYOPENDEPLOY/etc/jmsConfig.xml -stopServer"
             startDir="MYOPENDEPLOY/openjms">

<environment name="OPENJMS_CP" value="MYOPENDEPLOY/openjms/lib/openjms-0.7.2.jar"
/>
        <environment name="OPENJMS_CP"
value="${OPENJMS_CP}${path.separator}${java.class.path}" />
        <environment name="OPENJMS_RMI_PORTS" value="-DRmiPort1=${JMS_RMI1} -
DRmiPort2=${JMS_RMI2} -DRmiPort3=${JMS_RMI3} -DRmiPort4=${JMS_RMI4} -
DRmiPort5=${JMS_RMI5} -DRmiPort6=${JMS_RMI6} -DRmiPort7=${JMS_RMI7} -
DRmiPort8=${JMS_RMI8}" />

    </process>

    <jndiContext
initialContextFactory="org.exolab.jms.jndi.rmi.RmiJndiInitialContextFactory"
                 url="rmi://MYHOSTNAME:JMSRMIPORT/JndiServer">
        <connectionFactory factoryName="JmsTopicConnectionFactory">
            <connection
exceptionListener="com.interwoven.deploy.event.TExceptionListener">
                <property name="useLog" value="openDeployPublisherLog"/>
                <session transactional="false"
                         acknowledgeMode="CLIENT_ACKNOWLEDGE">
                  <publisher name="openDeployPublisher" topic="InterwovenEvents"/
>
                </session>
            </connection>
        </connectionFactory>
    </jndiContext>
</eventReportingConfiguration>
```

File Location

The host reporting configuration file resides in the following location in the host:

```
od-home/etc/eventReportingConfig.xml
```

You have the option of renaming this file. However, if you rename it, you must also update the file reference in the `eventReporting` element's `cfgPath` attribute value in the host's base server or receiver configuration file. See "Host Configuration File" on page 266 for more information.

The host reporting configuration file is installed with default settings allowing you to use the reporting feature with no additional modification required. However, as you use the reporting feature, you may want to change the settings to customize the reporting to your enterprise's particular requirements. The following sections describe different modifications you can make to the file.

Logging

The reporting feature generates it own log file. By default, this file resides in the following location:

```
od-home/log/publisher.log
```

You can configure the log entries and file location in the reporting configuration file through the `log` element:

```
<eventReportingConfiguration>
   <log
      name="reportingLog"
      path="C:\Interwoven\OpenDeployNG\eventlog\publisher.log"
      append="false" />
   ...
</eventReportingConfiguration>
```

The `log` element contains the following associated attributes:

- `name` — denotes the unique name of the `log` element. For example:

  ```
  name="reportingLog"
  ```

- `path` — specifies the absolute path to the log file. For example:

  ```
  path="od-home/eventlog/publisher.log"
  ```

- append — (optional) specify whether the file should be appended to (true) or truncated (false). If the value is true, new log entries are appended to the end of the existing log file. If the value is false, when OpenDeploy is started, the log file's existing entries are deleted, and replaced by new ones. The default value is true.

Administration Server Configuration for Reporting

The administration server contains a reporting management configuration file for use in displaying generated reports in the browser-based user interface.

```
<odReportingConfiguration hostName="ODSVR_HOSTNAME">

    <log name="openDeploySubscriberLog"
         path="ADMINSERVER_OD_DIR/log/MYHOSTNAME_subscriber.log"
         append="true"/>
    <log name="databaseLog"
         path="ADMINSERVER_OD_DIR/log/MYHOSTNAME_database.log"
         append="true"/>

    <database name="ReportingDB"
              className="org.hsqldb.jdbcDriver"
              connectionString="jdbc:hsqldb:ADMINSERVER_OD_DIR/db/
eventReporting.db">
        <property name="user" value="sa" />
        <property name="password" value="" />
    </database>

<!-- Repeat the odNode element for each OD base/receiver being subscribed to  -->
    <odNode host="ODSVR_HOSTNAME" port="JMSRMIPORT"/>

</odReportingConfiguration>
```

Upon installation, this configuration file requires little or no modification to work. However, for production use it is strongly recommended that you use your own JDBC-compliant database rather than the demonstration database that is included. Refer to the *OpenDeploy Release Notes* for a list of certified databases.

You also must modify the configuration file to give additional hosts access to the reporting structure.

File Location

The reporting management configuration file resides in the following location:

```
admin-home/odadmin/config/adminEventReportingConfig.xml
```

This is a fixed file that cannot be renamed or relocated.

Reporting Server Database

The reporting server requires database software to manage the reporting. By default, the reporting server software is installed with a Hypersonic database. However, this database is included for demonstration purposes and is not sufficiently powerful for most enterprise requirements.You are strongly encouraged to use your own JDBC-compliant database instead. Refer to the *OpenDeploy Release Notes* for a list of certified databases you can use.

Configuring Your Database

In the reporting management configuration file, the database is specified by the `database` element:

```
<odReportingConfiguration ...>
    ...
    <database
        name="ReportingDB"
        className="org.hsqldb.jdbcDriver"
        connectionString="jdbc:hsqldb:C:\Interwoven\AdminServer\db\
            eventReporting.db"
        >
        <property name="user" value="sa" />
        <property name="password" value="123ABC" />
    </database>
    ...
<odReportingConfiguration>
```

The `database` element contains the following required attributes that you must configure:

- `name` — denotes the unique name of the `database` element, for example:

```
name="ReportingDB"
```

- className — specify the class name of the JDBC driver class to load. For example:

  ```
  className="org.hsqldb.jdbcDriver"
  ```

- connectionString — specifies the JDBC connection string (URL) to use to connect to the database. For example:

  ```
  connectionString="jdbc:hsqldb:C:\Interwoven\AdminServer\db\
  eventReporting.db"
  ```

  Refer to java.sql.DriverManager.getConnection() in the Java API documentation for more information.

Specifying the Database User Name and Password

Depending on the type of database you are using, you might need to configure a database user name and password. You can specify both of these items using property elements within the database element, for example:

```
<database ...>
   <property name="user" value="sa" />
   <property name="password" value="123ABC" />
</database>
```

The property element resides within the database element, and has the following attributes:

- name — specifies the classification of the property element, for example:

  name="user" *or*

  name="password"

- value — specifies the value, for example:

  value="sa" *or*

  value="123ABC"

The need for a user name and password is JDBC driver dependent. Any property elements specified are passed to the JDBC driver as properties that it may use for establishing the connection or setting driver options. Only one user name and password is supported for the database. Refer to your database documentation to determine whether a user name and password are required.

Resetting the Hypersonic Database

In some cases, for example during demonstrations of the reporting feature, you might want to reset the default Hypersonic database. Resetting the database clears it of existing reporting information and allows the reporting feature to have a fresh start.

To reset the Hypersonic database, follow these steps:

1. Stop the administration server service or daemon. See "Stopping OpenDeploy" on page 53 for more information.
2. Navigate to the following location:

   ```
   admin-home/odadmin/db
   ```

3. Enter the following commands in order at the prompt:

   ```
   ./iwoddbtool -sql dropODEvents_hSql.sql
   ./iwoddbtool -sql ODEvents_hSql.sql
   ./iwoddbtool -sql quickreportlist_hSql.sql
   ```

4. Restart the administration server service or daemon. See "Starting OpenDeploy" on page 47 for more information.

Using Your Own Database

By default, OpenDeploy installs the Hypersonic database for use with the reporting server software. Upon installation, the Hypersonic database is initialized and ready to use. However, this database is included for demonstration purposes, and is probably not sufficiently powerful for most enterprise reporting requirements.

You can configure the reporting server to use your own database. Several databases have been certified for use with the reporting server software, and customized initialization scripts for those databases are included with the OpenDeploy software. Refer to the *OpenDeploy Release Notes* for a list of those certified databases and initialization scripts.

Alternatively, you can use a non-certified JDBC-compliant database with the reporting server. However, you must provide your own initialization script for that database. You can use one of the provided initialization scripts provided as a starting basis to develop your own initialization script. certified list.

To configure your own reporting server database, follow these steps:

1. Stop the administration server service or daemon. See "Stopping OpenDeploy" on page 53 for more information.

2. Open the reporting management configuration file using your favorite text or XML editor. The file resides in the following location:

   ```
   admin-home/odadmin/config/adminEventReportingConfig.xml
   ```

3. Edit the `database` element and its associated elements and attributes to reference and configure the database you want to work in conjunction with the reporting server. See "Configuring Your Database" on page 271

4. Save you changes and close the file.

5. Copy the JDBC driver `.jar` files associated with your database to the following location:

   ```
   admin-home/httpd/iwwebapps/opendeploy/WEB-INF/lib
   ```

6. Use your database's tools to create and initialize the tables.

   If you are using a certified database, you can run the initialization scripts provided with OpenDeploy. Refer to the *OpenDeploy Release Notes* for a list of the certified databases and the associated initialization scripts.

   If you are using a non-certified JDBC-compliant database, you must create your own initialization script. Refer to Chapter 12, "Reporting Server Database Schema" on page 229 in the *OpenDeploy Installation and Reference Guide* for a listing and description of the database schema to which the initialization script initializes. A file containing this schema also resides at the following location:

   ```
   admin-home/db/odreportschemas.txt
   ```

   You also can use the following files as guides:

   – `admin-home/db/ODEvents.sql` — defines the reporting schema. This is a base version that is not customized for any particular database.

   – `admin-home/db/quickreportlist.sql` — contains the definitions of the default quick reports. This is a base version that is not customized for any particular database.

7. Restart the administration server service or daemon. See "Starting OpenDeploy" on page 47 for more information.

Adding Hosts to the Reporting Environment

After installing the reporting server software as part of the administration package, you must configure the software to receive published events from each OpenDeploy host in the reporting environment.The reporting management configuration file (adminEventReportingConfig.xml) must contain an occurrence of the odNode element for each host:

```
<odReportingConfiguration ...>
   ...
   <odNode host="mars" port="9173" />
</odReportingConfiguration>
```

The odNode element has the following associated attributes:

- host — specifies the fully qualified DNS host name or IP address of the host.
- port — specifies the port number used by the host.
- unsubscribed — (optional) indicates whether (true) or not (false) the host had been previously subscribed into the reporting environment and should now be unsubscribed. Unsubscribing cleans up resources which might not be released if the odNode element entry is removed. Default value is false.

Any additional hosts you want to include in the reporting environment would need to have their own associated odNode elements added to the reporting management configuration file in the same manner. For example:

```
<odReportingConfiguration ...>
   ...
   <odNode host="mars" port="9173 />
   <odNode host="venus" port="9174 />
   <odNode host="jupiter" port="9175 />
</odReportingConfiguration>
```

Logging

The reporting management configuration file generates it own log file. By default, this file resides in the following location:

```
admin-home/log/host-name_subscriber.log
```

where *host-name* is the host of the administration server software.

Logging for the reporting management configuration file is configured similarly to the host reporting configuration file, for example:

```
<log
   name="openDeploySubscriberLog"
   path="admin-home/log/mars_subscriber.log"
   append="true"
   />
```

See "Logging" on page 269 for descriptions on logging behavior and descriptions of the attributes.

Custom Reports

Custom reports are reports that provide information on deployments. These reports are accessed through the browser-based user interface. You can also download them to your local host, and open them using other applications.

Custom reports have a fixed structure that provides the basic information that typical OpenDeploy users want without having to build a complete reporting structure yourself. However, you can apply a variety of search criteria to this structure to refine the report to the deployment information and timeframe you want.

When generated, a custom report contains the following specific type of reports:

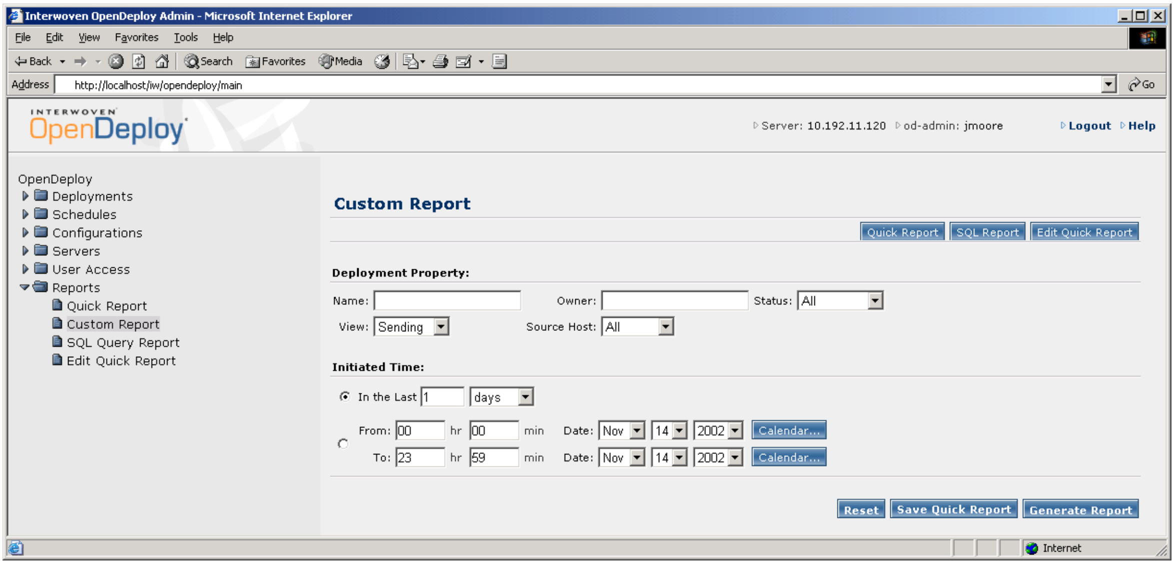
- Deployment report — displays information regarding the overall deployment.
- Deployment leg report — displays information regarding the deployment of files from a source host to a specific target, either as a single target deployment, a fan-out deployment, or a multi-tiered deployment.

- Manifest report — displays information on the status of each item deployed in a specific leg of the deployment.

You can access a particular deployment leg report from within the deployment report, and a particular manifest report from the deployment leg report.

Configuring Custom Report Queries

Custom reports are generated based on the user-defined custom report query. This query determines the search criteria used to determine the contents of the report.Custom report queries are created in the Custom Report window (Figure 50).



Address: http://localhost/iw/opendeploy/main

Server: 10.192.11.120 od-admin: jmoore

Custom Report

Deployment Property:

Name: Owner: Status: All

View: Sending Source Host: All

Initiated Time:

● In the Last 1 days

○ From: 00 hr 00 min Date: Nov 14 2002

To: 23 hr 59 min Date: Nov 14 2002

Figure 50: Custom Report Window

The Custom Report window contains a variety of items with which you can create a custom report query including the following search criteria:

- Deployment name.
- User name of the individual starting the deployment.

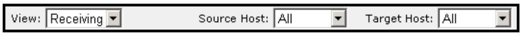
- Whether the search should include only deployments that are completed, in-progress, or failed, or whether it should include all deployments.
- Whether the report should cover OpenDeploy hosts sending or receiving deployments.
- Source and target (if applicable) hosts of the deployment.
- Start and end timeframe covered by the report

Creating Custom Report Queries

To create a custom report query, following these steps:

1. Select **Reports > Custom Report** to display the Custom Report window.
2. Enter the name of the deployment in the **Name** box if you want to limit the report's coverage. If you leave the **Name** box empty, all applicable deployments are included in the report.
3. Enter the appropriate user name in the **Owner** box if you want to limit the report to those deployments started by that individual. If you leave the **Owner** box empty, all applicable deployments started by any user are included in the report.
4. Select one of the following status types for the deployments from the **Status** list:
   - **Completed**
   - **In-progress**
   - **Failed**

   You can also select **All** to include all three status types in the report.
5. Select one of the following options from the **View** list:
   - **Sending** — includes information from hosts sending deployments.
   - **Receiving** — includes information from hosts receiving deployments. If you select **Receiving**, the **Target Host** list appears in the window (Figure 51).



View: Receiving Source Host: All Target Host: All

Figure 51: Target Host List When Receiving Is Selected

6. Select the button associated with the following timeframe option you want to apply to the custom report, and complete the information required for that option:
   - **In the Last** — enter a number and select the corresponding measure of time (hour, day, week, month) that the report will cover.
   - **From/To** — enter the hours and minutes and select the dates from start to end that will be covered by the report.You can select the **Calendar** buttons to display a calendar tool to select the days.

You can click **Reset** at any time while creating a custom report to delete the values you entered and start again.

After you have completed creating the custom report query, you can generate the report. You can also save the custom report query as a quick report if you want to run the report in the future without having to recreate it. See “Generating Custom Reports” on page 280 and “Saving Custom Reports as a Quick Reports” on page 285 for more information.

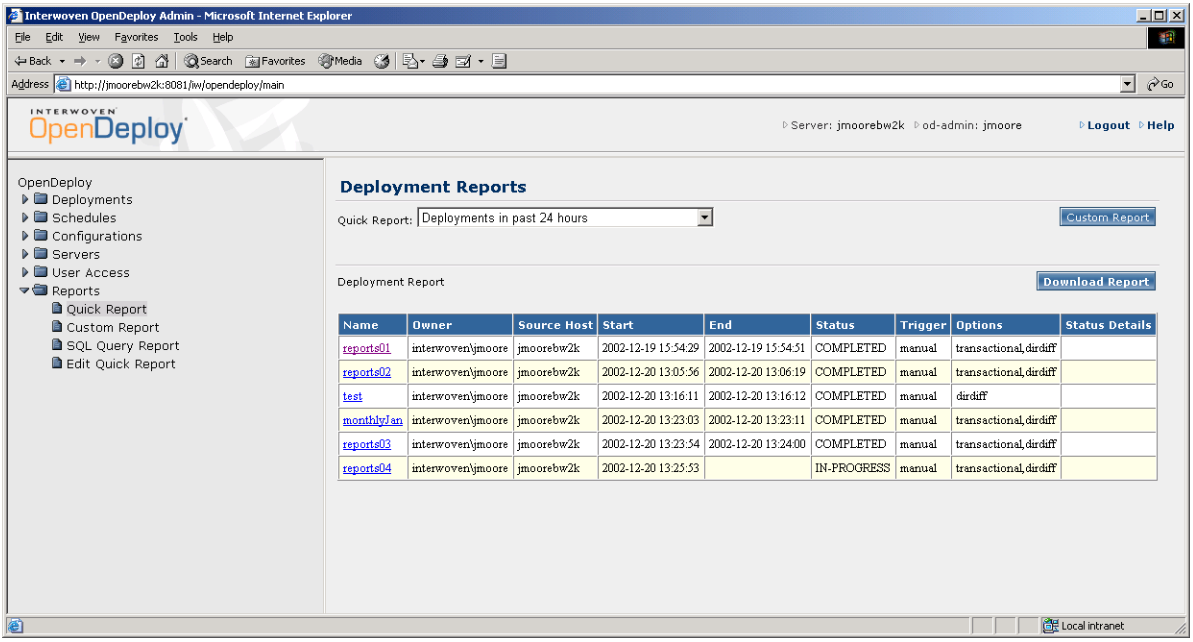
Exporting Custom Report Queries

After you create your custom report query, you can export it into the SQL Query Report window, where you can further customize it. SQL query reports provide a greater level of flexibility to reporting than is available with custom reports.

To apply a custom report to an SQL query, click **SQL Report** in the Custom Report window to display the SQL Query Reports window. The query information from the custom report is automatically imported into the SQL query displayed in the SQL Query Reports window. See “SQL Query Reports” on page 286 for more information.

Generating Custom Reports

You can generate a custom report after you have configured it by clicking **Generate Report** in the Custom Report window. The Deployment Reports window is displayed (Figure 52), containing the generated report.



http://jmoorebw2k:8081/iw/opendeploy/main

Server: jmoorebw2k   od-admin: jmoore

Deployment Reports

Quick Report: Deployments in past 24 hours

Deployment Report

Name	Owner	Source Host	Start	End	Status	Trigger	Options	Status Details
reports01	interwoven\jmoore	jmoorebw2k	2002-12-19 15:54:29	2002-12-19 15:54:51	COMPLETED	manual	transactional,dirdiff	
reports02	interwoven\jmoore	jmoorebw2k	2002-12-20 13:05:56	2002-12-20 13:06:19	COMPLETED	manual	transactional,dirdiff	
test	interwoven\jmoore	jmoorebw2k	2002-12-20 13:16:11	2002-12-20 13:16:12	COMPLETED	manual	dirdiff	
monthlyJan	interwoven\jmoore	jmoorebw2k	2002-12-20 13:23:03	2002-12-20 13:23:11	COMPLETED	manual	transactional,dirdiff	
reports03	interwoven\jmoore	jmoorebw2k	2002-12-20 13:23:54	2002-12-20 13:24:00	COMPLETED	manual	transactional,dirdiff	
reports04	interwoven\jmoore	jmoorebw2k	2002-12-20 13:25:53		IN-PROGRESS	manual	transactional,dirdiff	

Figure 52: Deployment Report Window

Preconfigured Reports

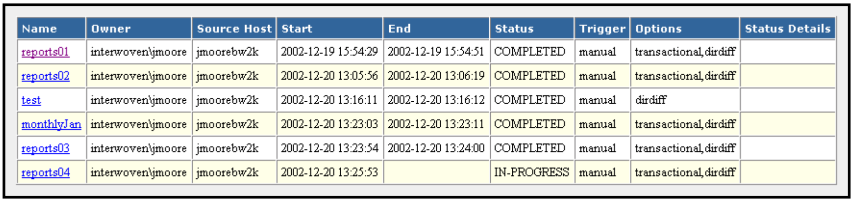
OpenDeploy includes the following preconfigured reports, known as *quick reports*, that are available for generation at any time:

- Deployments in the past 24 hours
- Sender completed deployments in past 24 hours
- Sender failed deployment in past 24 hours

You can also save any custom report query you create as a quick report and generate it at a later time. See “Saving Custom Reports as a Quick Reports” on page 285 for more information.

Deployment Report Structure

Deployment reports (Figure 53) provide general information on the overall deployment. These are the reports initially displayed when you select a custom report.



Name	Owner	Source Host	Start	End	Status	Trigger	Options	Status Details
reports01	interwoven\jmoore	jmoorebw2k	2002-12-19 15:54:29	2002-12-19 15:54:51	COMPLETED	manual	transactional,dirdiff	
reports02	interwoven\jmoore	jmoorebw2k	2002-12-20 13:05:56	2002-12-20 13:06:19	COMPLETED	manual	transactional,dirdiff	
test	interwoven\jmoore	jmoorebw2k	2002-12-20 13:16:11	2002-12-20 13:16:12	COMPLETED	manual	dirdiff	
monthlyJan	interwoven\jmoore	jmoorebw2k	2002-12-20 13:23:03	2002-12-20 13:23:11	COMPLETED	manual	transactional,dirdiff	
reports03	interwoven\jmoore	jmoorebw2k	2002-12-20 13:23:54	2002-12-20 13:24:00	COMPLETED	manual	transactional,dirdiff	
reports04	interwoven\jmoore	jmoorebw2k	2002-12-20 13:25:53		IN-PROGRESS	manual	transactional,dirdiff	

Figure 53: Deployment Report

Deployment reports are displayed in table format. Each column represents a category of information in the report:

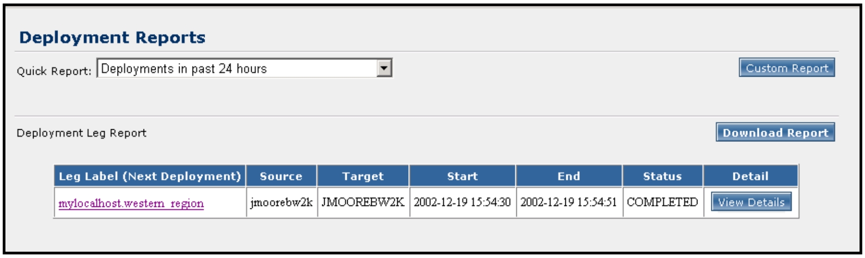
- **Name** — displays the name and instance (if appropriate) of the deployment. This name is a link, which when clicked, displays an additional report on each leg of the selected deployment.
- **Owner** — displays the user name of the user who ran the deployment.
- **Source Host** — displays the name of the source host. This name can be a fully qualified DNS host name or an IP address.
- **Start** — displays the start time of the deployment, using the following format:

  ```
  YYYY-MM-DD hh:mm:ss
  ```
- **End** — displays the end time of the deployment, using the following format:

  ```
  YYYY-MM-DD hh:mm:ss
  ```
- **Status** — indicates whether the deployment has completed, failed, or has been skipped.
- **Trigger** — displays how the deployment was started, such as manually, or by schedule.
- **Options** — displays information about the deployment type and features.
- **Status Details** — displays additional information as appropriate.

Deployment Leg Report

Deployment leg reports (Figure 54) provide information on each leg of a specific deployment.



Deployment Reports

Quick Report: Deployments in past 24 hours

Deployment Leg Report

Leg Label (Next Deployment)	Source	Target	Start	End	Status	Detail
mylocalhost.western_region	jmoorebw2k	JMOOREBW2K	2002-12-19 15:54:30	2002-12-19 15:54:51	COMPLETED	View Details

Figure 54: Deployment Leg Report

Each deployment leg can represent a deployment of content from a source host to a target host, as in the case of a single target or fan-out deployment, or it can represent the deployment of content from one tier to another tier in a multi-tier deployment.

You can display the deployment leg report for a specific deployment by clicking that deployment's link under the **Name** column in the deployment report table. Deployment leg reports contain the following information:

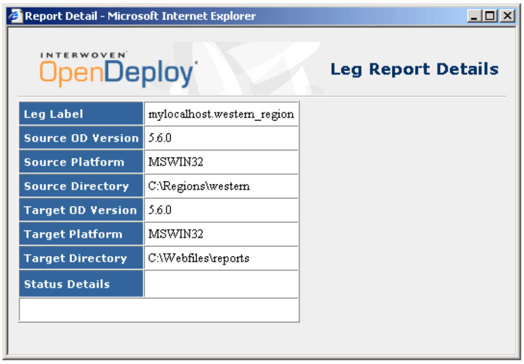
- **Leg Label (Next Deployment)** column — displays the deployment leg name, which is a combination of the target node name and the deployment definition name, as a link. When clicked, the Manifest Report for that leg is displayed.
- **Source** column — displays the source host of the deployment leg.
- **Target** column — displays the target host of the deployment leg.
- **Start** column — displays the start time of the deployment leg, using the following format:

  ```
  yyyy-mm-dd hh:mm:ss
  ```

- **End** column — displays the end time of the deployment leg, using the following format:

  ```
  yyyy-mm-dd hh:mm:ss
  ```

- **Status** column — displays whether the deployment leg was completed or failed.
- **View Details** button — click to display the Leg Report Details window (Figure 55), which contains information on the deployment leg, including the leg name, source and target platforms, and source and target file locations.

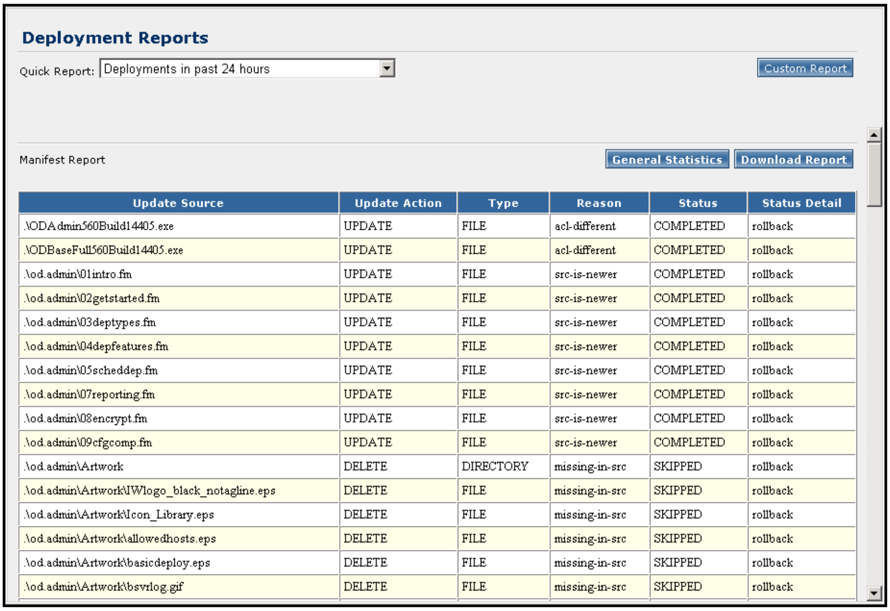


Leg Label	mylocalhost.western_region
Source OD Version	5.6.0
Source Platform	MSWIN32
Source Directory	C:\Regions\western
Target OD Version	5.6.0
Target Platform	MSWIN32
Target Directory	C:\Webfiles\reports
Status Details	

Figure 55: Leg Report Details Window

Manifest Report

Deployment manifest reports (Figure 56) provide information on the content associated with a specific deployment leg.



Update Source	Update Action	Type	Reason	Status	Status Detail
.\ODAdmin560Build14405.exe	UPDATE	FILE	acl-different	COMPLETED	rollback
.\ODBaseFull560Build14405.exe	UPDATE	FILE	acl-different	COMPLETED	rollback
.\od.admin\01intro.fm	UPDATE	FILE	src-is-newer	COMPLETED	rollback
.\od.admin\02getstarted.fm	UPDATE	FILE	src-is-newer	COMPLETED	rollback
.\od.admin\03deptypes.fm	UPDATE	FILE	src-is-newer	COMPLETED	rollback
.\od.admin\04depfeatures.fm	UPDATE	FILE	src-is-newer	COMPLETED	rollback
.\od.admin\05scheddep.fm	UPDATE	FILE	src-is-newer	COMPLETED	rollback
.\od.admin\07reporting.fm	UPDATE	FILE	src-is-newer	COMPLETED	rollback
.\od.admin\08encrypt.fm	UPDATE	FILE	src-is-newer	COMPLETED	rollback
.\od.admin\09cfgcomp.fm	UPDATE	FILE	src-is-newer	COMPLETED	rollback
.\od.admin\Artwork	DELETE	DIRECTORY	missing-in-src	SKIPPED	rollback
.\od.admin\Artwork\IWlogo_black_notagline.eps	DELETE	FILE	missing-in-src	SKIPPED	rollback
.\od.admin\Artwork\Icon_Library.eps	DELETE	FILE	missing-in-src	SKIPPED	rollback
.\od.admin\Artwork\allowedhosts.eps	DELETE	FILE	missing-in-src	SKIPPED	rollback
.\od.admin\Artwork\basicdeploy.eps	DELETE	FILE	missing-in-src	SKIPPED	rollback
.\od.admin\Artwork\bsvrlog.gif	DELETE	FILE	missing-in-src	SKIPPED	rollback

Figure 56: Deployment Manifest Report

The report provides the information on each file and directory deployed:

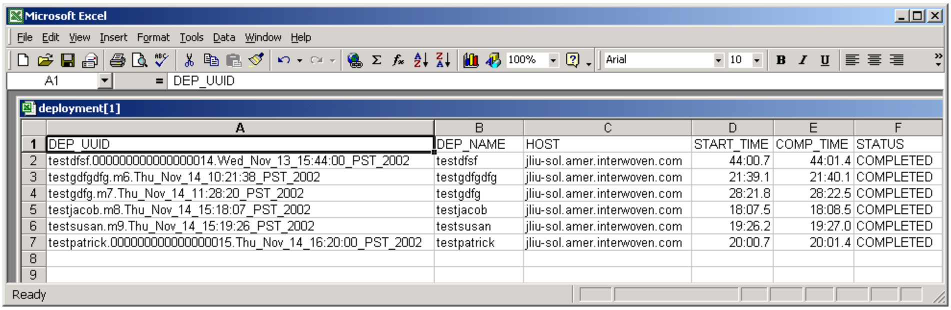
- **Update Source** column — displays the name of the status file associated with the deployment. The file resides in the following location:

  od-home/tmp

- **Update Action** column — displays whether the deployed item was new, updated, or deleted.
- **Type** column — displays whether the deployed item was a file, a directory, or a link.
- **Reason** column — displays the reason the item was acted upon.
- **Status** column — displays whether the deployed item was completed, failed, or skipped.
- **Status Detail** column — displays additional information as appropriate.

Downloading Custom Reports

You can download a generated custom report to your local host or another computer on the network. The saved file is a comma-delimited file (CSV) that can be viewed and used by another application, such as a database or a text editor. Figure 57 shows a custom report being displayed in Microsoft Excel.



	A	B	C	D	E	F
1	DEP_UUID	DEP_NAME	HOST	START_TIME	COMP_TIME	STATUS
2	testdfsf.00000000000000014.Wed_Nov_13_15:44:00_PST_2002	testdfsf	jliu-sol.amer.interwoven.com	44:00.7	44:01.4	COMPLETED
3	testgdfgdfg.m6.Thu_Nov_14_10:21:38_PST_2002	testgdfgdfg	jliu-sol.amer.interwoven.com	21:39.1	21:40.1	COMPLETED
4	testgdfg.m7.Thu_Nov_14_11:28:20_PST_2002	testgdfg	jliu-sol.amer.interwoven.com	28:21.8	28:22.5	COMPLETED
5	testjacob.m8.Thu_Nov_14_15:18:07_PST_2002	testjacob	jliu-sol.amer.interwoven.com	18:07.5	18:08.5	COMPLETED
6	testsusan.m9.Thu_Nov_14_15:19:26_PST_2002	testsusan	jliu-sol.amer.interwoven.com	19:26.2	19:27.0	COMPLETED
7	testpatrick.00000000000000015.Thu_Nov_14_16:20:00_PST_2002	testpatrick	jliu-sol.amer.interwoven.com	20:00.7	20:01.4	COMPLETED
8						
9						

Figure 57: Generated Custom Report Opened in Microsoft Excel

Downloading a generated custom report allows you to modify the report, copy and paste portions into other documents, or use the application to save it under a different format.

To download a generated custom, follow these steps:

1. Click **Download Report** in the Deployment Report window. A message window appears prompting whether you want to open or save the report file.
2. Click **Save**. You are prompted to locate where you want to save the file and under what file name.
3. Navigate to the location where you want the file to be saved, and provide a file name.
4. Save the file.

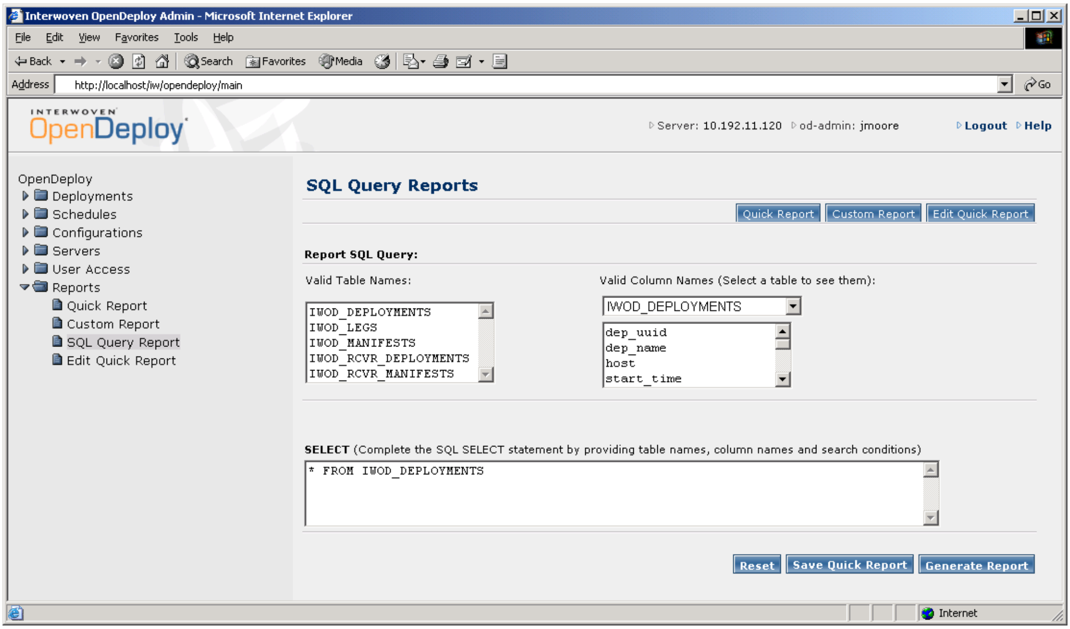
Saving Custom Reports as a Quick Reports

You can save any custom report query as a quick report, where you can access and generate the report at anytime. Click **Save Quick Report** when you create your custom report query. You will be prompted to name the report query. That named report is subsequently listed among the quick reports in the Deployment Reports window. See “Quick Reports” on page 290 for more information.

SQL Query Reports

SQL query reports allow you to design and compose your own reporting query when the custom report feature does not offer enough flexibility. Using SQL, you can compose a single search query that can include individual columns from a variety of available tables to create a completely customized report. SQL query reports can be saved as quick reports for future usage, the same as with custom reports.

SQL query reports are created in the SQL Query Reports window (Figure 58).



Address: http://localhost/iw/opendeploy/main

Server: 10.192.11.120 od-admin: jmoore

SQL Query Reports

Report SQL Query:

Valid Table Names:

```
IWOD_DEPLOYMENTS
IWOD_LEGS
IWOD_MANIFESTS
IWOD_RCVR_DEPLOYMENTS
IWOD_RCVR_MANIFESTS
```

Valid Column Names (Select a table to see them): IWOD_DEPLOYMENTS

```
dep_uuid
dep_name
host
start_time
```

SELECT (Complete the SQL SELECT statement by providing table names, column names and search conditions)

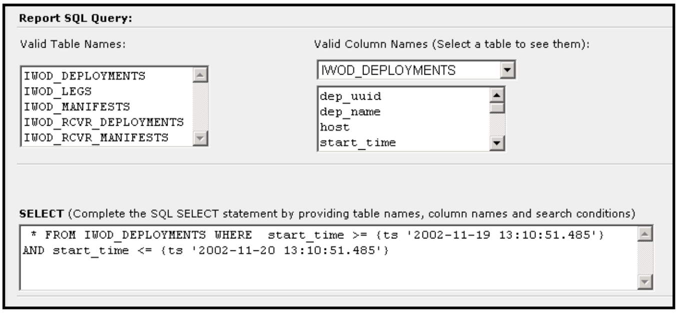
```
* FROM IWOD_DEPLOYMENTS
```

Figure 58: SQL Query Reports Window

The SQL Query Reports window is displayed when you select **Reports > SQL Query Report** in the browser-based user interface. This window is also displayed when you click the **SQL Report** button in the Custom Report window, allowing you to import your custom report into the SQL Report window. Here, you can further customize it by adding user-defined tables, columns, and search terms.

SQL Report Queries

The SQL Query Reports window contains the information required for you to create your SQL report query (Figure 59).



Report SQL Query:

Valid Table Names:

IWOD_DEPLOYMENTS
IWOD_LEGS
IWOD_MANIFESTS
IWOD_RCVR_DEPLOYMENTS
IWOD_RCVR_MANIFESTS

Valid Column Names (Select a table to see them):

IWOD_DEPLOYMENTS

dep_uuid
dep_name
host
start_time

SELECT (Complete the SQL SELECT statement by providing table names, column names and search conditions)

```
 * FROM IWOD_DEPLOYMENTS WHERE  start_time >= {ts '2002-11-19 13:10:51.485'}
AND start_time <= {ts '2002-11-20 13:10:51.485'}
```

Figure 59: Composing SQL Query Scripts

The **Valid Table Nam**e list contains those tables whose individual columns are valid and available for inclusion in an SQL search query. The **Valid Column Name** list contains those columns associated with the table selected in the **Valid Table Name** list.

Both of these lists are for information purposes only. You can use the valid table and column information provided in these lists in your SQL query script.

You can create a single SELECT query in the **SELECT** box. In this box you can enter those valid table and column names, along with the appropriate search conditions. You can copy and paste selected items into the **SELECT** box, or use drag-and-drop to build your query.

Creating SQL search queries requires some level of SQL expertise. If you are not comfortable with composing SQL scripts, you should use the custom reports feature instead. See “Custom Reports” on page 276 for more information.

Case Sensitivity

Case sensitivity in SQL query statements is handled differently for various platforms and RDBMS vendors.You should be aware of that when writing your own customer queries. Refer to your database documentation for more information.

Creating SQL Queries

To create an SQL query, follow these steps:

1. Select **Reports > SQL Query Report** to display the SQL Query Reports window.
2. Review those available tables in the **Valid Table Names** list.
3. Review those available columns that correspond to a particular table by selecting the table from the table list. The corresponding columns are displayed in the **Valid Column Names** list.
4. Compose a single search SQL query by entering the valid tables, columns, and search conditions in the **SELECT** text box.

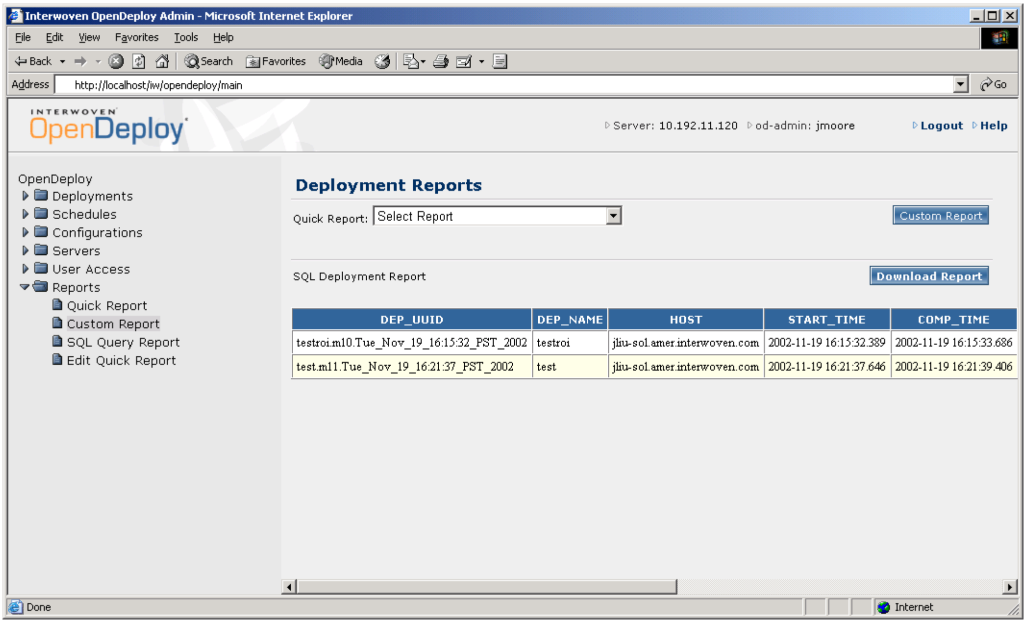
   You can compose your query by copying and pasting a selected table or column name into the **SELECT** text box, or by using drag-and drop.

You can click **Reset** at any time while creating an SQL query report to delete the values you entered and start again.

After you have completed creating the SQL report query, you can generate the report. You can also save the SQL query report as a quick report if you want to run the report in the future without having to recreate it.

Generating SQL Query Reports

You can generate an SQL query report after you have created the report query by clicking **Generate Report** in the SQL Query Reports window. The Deployment Reports window is displayed (Figure 52), containing the generated report.


Interwoven OpenDeploy Admin - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/iw/opendeploy/main
Go
INTERWOVEN OpenDeploy
Server: 10.192.11.120 od-admin: jmoore
Logout Help
OpenDeploy
Deployments
Schedules
Configurations
Servers
User Access
Reports
Quick Report
Custom Report
SQL Query Report
Edit Quick Report
Quick Report: Select Report
Custom Report
Download Report
Done
Internet

Deployment Reports

SQL Deployment Report

DEP_UUID	DEP_NAME	HOST	START_TIME	COMP_TIME
testroi.m10.Tue_Nov_19_16:15:32_PST_2002	testroi	jliu-sol.amer.interwoven.com	2002-11-19 16:15:32.389	2002-11-19 16:15:33.686
test.m11.Tue_Nov_19_16:21:37_PST_2002	test	jliu-sol.amer.interwoven.com	2002-11-19 16:21:37.646	2002-11-19 16:21:39.406

Figure 60: Deployment Report Windows Displaying Generated SQL Query Report

Downloading an SQL Query Report

You can download a generated SQL query report to your local host or another computer on the network. The saved file is a comma-delimited file that can be viewed and used by another application, such as a database or a text editor. The procedure is the same as for custom reports. See "Downloading Custom Reports" on page 285 for more information.

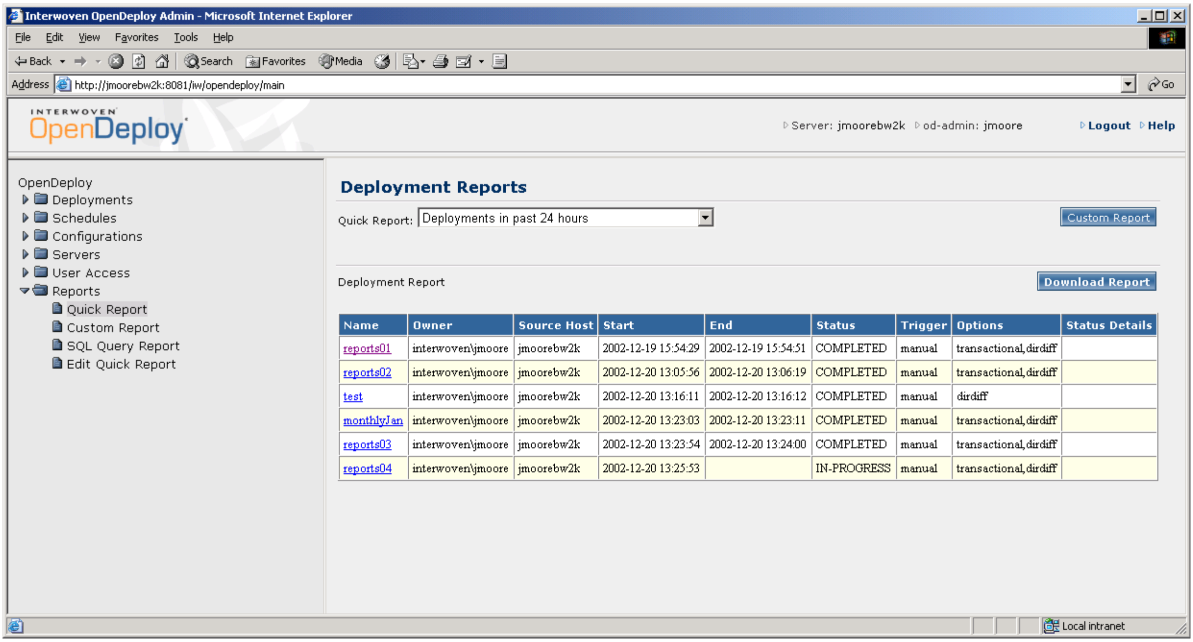
Saving an SQL Query Report as a Quick Report

You can save any SQL query report as a quick report, where you can access and generate the report at anytime. Click **Save Quick Report** when you create your SQL query report. You will be prompted to name the report. That named report is subsequently listed among the quick reports in the Deployment Reports window. The following section describes quick reports.

Quick Reports

You can save any custom or SQL query report you create as a quick report. The report query is saved and can be accessed and run at any time with no additional report configuration required. If you plan to run certain reports on a regular basis, you should consider saving them as quick reports.

Quick reports are displayed in the Deployment Reports window (Figure 61).



Address http://jmoorebw2k:8081/iw/opendeploy/main

Server: jmoorebw2k od-admin: jmoore

Deployment Reports

Quick Report: Deployments in past 24 hours

Deployment Report

Name	Owner	Source Host	Start	End	Status	Trigger	Options	Status Details
reports01	interwoven\jmoore	jmoorebw2k	2002-12-19 15:54:29	2002-12-19 15:54:51	COMPLETED	manual	transactional,dirdiff	
reports02	interwoven\jmoore	jmoorebw2k	2002-12-20 13:05:56	2002-12-20 13:06:19	COMPLETED	manual	transactional,dirdiff	
test	interwoven\jmoore	jmoorebw2k	2002-12-20 13:16:11	2002-12-20 13:16:12	COMPLETED	manual	dirdiff	
monthlyJan	interwoven\jmoore	jmoorebw2k	2002-12-20 13:23:03	2002-12-20 13:23:11	COMPLETED	manual	transactional,dirdiff	
reports03	interwoven\jmoore	jmoorebw2k	2002-12-20 13:23:54	2002-12-20 13:24:00	COMPLETED	manual	transactional,dirdiff	
reports04	interwoven\jmoore	jmoorebw2k	2002-12-20 13:25:53		IN-PROGRESS	manual	transactional,dirdiff	

Figure 61: Deployment Reports Window

The **Quick Report** list contains all those quick reports that you can access and display. The following preconfigured quick reports are also included for your use with no additional configuration required:

- **Today's Sender Failed Deployments** — displays a report of failed deployments over the previous 24 hour period.
- **Today's Deployments** — displays a report of all deployments over the previous 24 hour period.
- **Today's Sender Completed Deployments** — displays a report of all completed deployments over the previous 24 hour period.

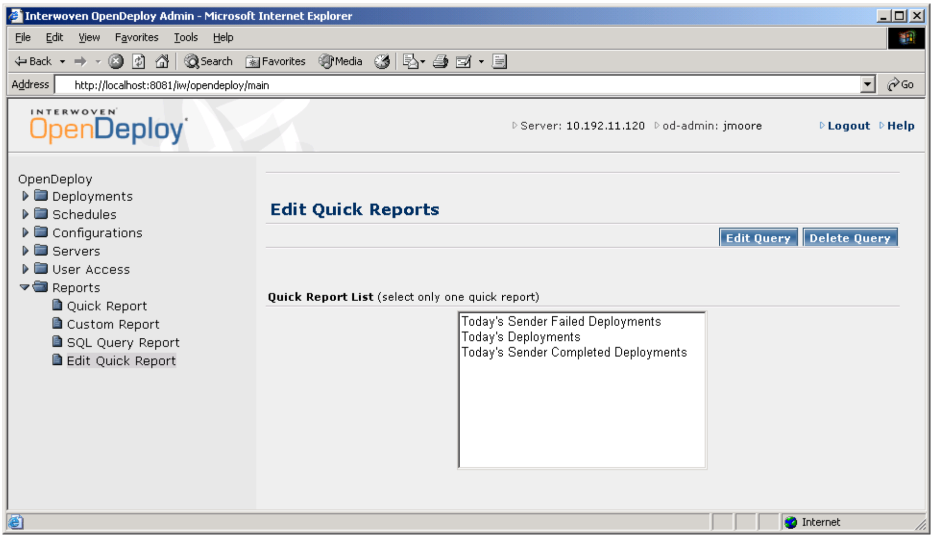
Selecting an entry from the **Quick Report** list runs that report and displays the results in the Deployment Reports window. You can also download the report to your local host by clicking **Download Report**. See the sections on custom and SQL query reports for instructions on how to use this feature for those types of reports.

Adding New Entries to Quick Report List

You can add any custom or SQL query report you create to the **Quick Report** list by clicking the **Save Quick Report** button in the respective Custom Report or SQL Query Reports window at the time of creation.

Editing Existing Entries

You can edit a custom or SQL query listed in the **Quick Report** list through the Edit Quick Reports window (Figure 62).



Today's Sender Failed Deployments
Today's Deployments
Today's Sender Completed Deployments

Figure 62: Edit Quick Reports Window

Select **Reports > Edit Quick Reports** to display the Edit Quick Reports window.

The Edit Quick Reports window lists all available quick reports, along with buttons to edit or delete the quick report you select. Selecting a quick report entry from the **Quick Report** list and clicking **Edit Query** will display the associated Custom Report or SQL Query Reports window, where you can modify the report. After making you changes, you can save the report under its existing name, or save it with a new name. You can also generate the report.

Deleting Entries

You can delete any existing quick report by opening the Edit Quick Reports window, selecting the quick report you want to delete from the **Quick Reports** list, and clicking **Delete**. After deleting a quick report, that report no longer appears in the **Quick Reports** list, and is no longer available for use.

Chapter 8

Encryption

OpenDeploy provides two methods of encryption:

- Weak (40-bit) symmetric key file-based encryption
- Secure data transfer using Secure Sockets Layer-based (SSL) encryption

These types of encryption are mutually exclusive and cannot be used in conjunction with one another. Be sure not to include attributes for both types of encryption in the same configuration.

Encryption can be specified both at the OpenDeploy base server and receiver level, and at the individual deployment configuration level. Encryption settings specified in the deployment configuration level will automatically override any encryptions settings in the server configuration.

Encryption is not supported by the EasyDeploy base server software. To use encryption, you must upgrade to the full-feature base server software.

Symmetric Key Encryption

OpenDeploy provides 40-bit encryption support for content transfers through referencing an encryption algorithm key file specified in the base server or receiver configuration file. OpenDeploy symmetric key deployment provides basic encryption support with minimal performance impact on content deployment. However, symmetric 40-bit encryption is breakable by brute force attack with a modest amount of computing power and is potentially vulnerable to unauthorized users with the same symmetric key who can intercept data in transit.

Configuring OpenDeploy for Symmetric Encryption

The path and filename can be specified in the `keyFile` attribute of the `localNode` element in either the base server, receiver, or deployment configuration files. If you configure your base server or receiver configuration files for symmetric configuration, but do not specify encryption in the specific deployment configuration, then by default the deployment will use symmetric encryption. Whatever you specify for symmetric encryption in the deployment configuration will override any settings in the base server or receiver configuration files.

The `keyFile` attribute specifies the absolute path to the symmetric key. Here is an example of the OpenDeploy server *mars* being configured for symmetric key encryption:

```
<localNode
   host="mars"
   keyFile="/local/OpenDeploy/conf/keyfile.txt"
   />
```

Using Symmetric Encryption with Reverse Deployments

If you are performing a reverse deployment using symmetric key encryption, you must include the path to the symmetric key file residing on the reverse-source host (normally the receiving host in a forward deployment) in the deployment configuration. This is the same location as specified in the base server or receiver configuration file of the reverse-source host. This differs from a forward deployment, where the configuration would include the path to the key file where it resides on the source host. The deployment configuration must include the path syntax of the reverse-source host. The path to the symmetric key file is defined in the `keyFile` attribute of the `localNode` element.

In the following example, the source host *mars* has the base server software installed and is running on UNIX. The target host *venus* has the receiver software installed and is running on Windows. In a typical forward deployment using symmetric key encryption, the `localNode` element in the deployment configuration residing on *mars* would be:

```
<localNode
   host="mars"
   keyFile="/local/OpenDeploy/conf/keyfile.txt"
   />
```

and the `localNode` element in *venus*' receiver configuration file would be:

```
<localNode
   host="venus"
   keyFile="C:\encryption\keyfile.txt"
   />
```

In a reverse deployment involving these two hosts, the `localNode` element in the reverse deployment configuration would be:

```
<localNode
   host="mars"
   keyFile="C:\encryption\keyfile.txt"
   />
```

and the `localNode` element in the base server configuration file on *mars* would be:

```
<localNode
   host="mars"
   keyFile="/local/OpenDeploy/conf/keyfile.txt"
   />
```

Secure Data Transfer with SSL

OpenDeploy uses X509.v3 digital certificates and Secure Socket Layer (SSL) v3 for secure content transfer. OpenDeploy comes packaged with its own certificate authority, which should be used for certificate generation. A packaged script aids in the creation of the certificate authority and subsequent certificate generation.

This release of OpenDeploy supports DSA and RSA certificates, but has only been tested using the certificate authority that comes packaged with OpenDeploy. Using a certificate authority other then the one shipped with OpenDeploy may result in unexpected results.

Certificates that require a password to be used will not work with OpenDeploy.

The use of 168-bit encryption is available within the United States of America and most other countries. You can also set up OpenDeploy to accept multiple levels of encryption.

The following sections describe the process of creating digital certificates. This is a multi-step process that requires familiarizing yourself with the complete process before beginning any individual tasks. You should read this documentation completely before actually attempting this process.

Obtaining Additional SSL Information

You can find additional information on the SSL through the following Web site:

```
www.openssl.org
```

For more information about encryption and ciphers, consult a cryptography reference manual such as *Applied Cryptography* (Bruce Schneier, ISBN 0-471-11709-9).

Setting Up for SSL Private Keys and Certificates

Each peer host running OpenDeploy contains an SSL configuration within the base server or receiver configuration file's `localNode` element. OpenDeploy uses the OpenSSL implementation of the SSL. Setting up OpenDeploy involves the following tasks:

- Editing the certificate authority configuration file.
- Setting up the certificate authority (CA).
- Generating the certificates and keys for the base server.
- Generating the certificates and keys for the receiving nodes.
- Copying the certificate key/pair and the CA certificate to the other OpenDeploy nodes.
- Configuring the OpenDeploy base server configuration file (by default `odbase.xml`) if the base server host is to receive deployment using SSL.
- Configuring the receiver configuration file (by default `odrcvr.xml`) for SSL data transfer encryption.
- Configuring the deployment configuration for SSL data transfer encryption.

If you have one OpenDeploy sender and one OpenDeploy receiver, these tasks create two unique public and private key pairs that are signed by the same certificate authority. One key pair is copied to the source host. The other key pair and the CA's certificates are copied to the target host. These tasks are required regardless of what level of encryption you want to use. Either the source or target host has the ability to request a verification of the certificate authority before the deployment can occur. See "Configuring OpenDeploy for SSL Data Transfer Encryption" on page 306 for more information.

The *certificate authority* consists of a set of programs used to generate public and private key pairs and a database that contains state information. The programs are installed in the following location:

```
od-home/bin
```

The following table lists those files used for generating the certificate authority:

Windows	UNIX
`openssl.exe`	`openssl`
`openssl.cnf`	`openssl.cnf`
`CA.bat`	`CA.sh`

- The `openssl.exe` and `openssl` programs are command line tools for using the various cryptography functions of OpenSSL's cryptography library from the shell. See "Obtaining Additional SSL Information" on page 296 for more information.
- The `openssl.cnf` file is the configuration file for running the `openssl` utility.
- The `CA.bat` and `CA.sh` files are the wrapper scripts that are used both to create the certificate authority and to generate the certificates and private keys for OpenDeploy.

By default, the database is contained in the directory where the programs are run. If future public and private key pairs are created using a different certificate authority, OpenDeploy will not be able to deploy to or from a host with keys created by the original certificate authority. If a problem occurs during key generation, it is best to delete the created key and authority and start over.

Much of the state information that is used in creating the certificate/key pairs, including the certificate authority's certificate, is maintained in this directory. If future public and private key pairs are created using a different certificate authority, or the current authority is overwritten, OpenDeploy will not be able to deploy files using these certificates.

Setting Up the Certificate Authority

To set up the OpenSSL certificate authority, follow these steps:

1. Verify that the *od-home*`/bin` directory is included in the `PATH` environment variable.
   - UNIX — enter `env|grep PATH` at the prompt.
   - Windows — right click on the My Computer icon and select **Properties** from the pop-up menu to open the System Properties window. Next, select the **Advanced** tab to make it active. Finally, click **Environment Variables**. to open the Environmental Variables window. The current path in use is displayed in the **System variables** list.
2. Navigate to the following location:

   od-home`/bin`

3. Make a backup copy of `openssl.cnf` file, for example `openssl.cnf_orig`.
4. Open the `openssl.cnf` file using your favorite text editor.
5. Change the following line if you would like the certificate to last for longer than a year.

   ```
   default_days      =365          # how long to certify for
   ```

6. Change the default values for your own installation. These are located in the `[ req_distinguished_name ]` section of the file.
7. Ensure that the `RANDFILE` variable in `openssl.cnf` is set to:

   ```
   RANDFILE=.rnd
   ```

   When invoking the OpenSSL utilities, run them in *od-home*`/bin`, which is where the `openssl.cnf` file also resides.

8. Save and close the file.

9. Create a seed file (`*.rnd`) for the random number generator by performing the following steps:

   a. Enter the following command at the prompt:

   ```
   netstat -a > .rnd
   ```

   b. Move this `.rnd` file to the following location:

   ```
   od-home/bin
   ```

   As an alternative, you can copy any file sufficiently large into a `.rnd` file to make it a seed file. Log files are good example of random data for seeding. The key requirement is that the data used for seeding is truly random or very difficult to reproduce.

   OpenSSL uses a pseudo random number generator (PRNG) to generate public and private key pairs. The PRNG needs to be seeded with a satisfactory amount of genuine random data so it does not generate predictable keys.

   "Obtaining Additional SSL Information" on page 296 for more information on seeding methods.

10. Create the new certificate authority by entering the following command (depending on the platform) at the prompt:

    Windows — **`CA.bat -newca`** *or*

    UNIX — **`CA.sh -newca`** (press `Enter` at the prompt to create the new certificate authority.)

    By default, the certificate authority will have a life span of 365 days before expiring. If you want to specify another expiration date, you can append the command with the `-days` option and specify the number of days until expiration. See "Certificate Authority Expiration" on page 300 for more information.

    If the certificate authority already exists, the script will print harmless error messages regarding existing directories, and finish execution.

    Creating the certificate authority results in the following directory being created:

    ```
    od-home/bin/demoCA
    ```

    and copies the authority's certificate/key pair into it. A certificate authority can be initialized from a previously existing CA certificate, or it can be created as a completely new authority. The default method is to create a new authority.

11. Enter the appropriate information in response to the following prompt:

```
You are about to be asked to enter information that will be incorporated into
your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value,

If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [California]:
Locality Name (eg, city) [Sunnyvale]:
Organization Name (eg, company) []:Interwoven
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, YOUR name) []:Engineering Certificate Authority
Email Address []:
```

The prompts for country, state or province, and locality contain default values that you can accept, or you can enter your own information. The prompts for organization name, organizational unit name, common name, and email address are optional. You can either enter a value, or leave them blank by entering the value "`.`". All of the inputs to the prompts constitute the Distinguished Name.

The more unique values you provide for these optional prompts, the more effective the certificate authority will be. Each certificate authority you create must be unique from all other certificates. One method to ensure disparate Distinguished Names is by providing dissimilar values for the Common Name prompt.

You can begin the certificate authority process by deleting the directory containing the certificates. There is no penalty for this until you begin issuing certificates. You will not be able to use certificates that have the same Distinguished Name as the certificate authority. You will invalidate all certificates signed by a particular certificate authority by deleting its default directory.

Certificate Authority Expiration

By default, any certificate authority you create has a life span of 365 days before it expires. However, you can specify another expiration period at the time of creation if you want by appending the `CA.bat newca` or `CA.sh newca` command with the `-days` option and a number representing the number of days the certificate authority is valid before expiring.

For example, if you wanted to specify a certificate authority expiration date of 200 days after creation, you would enter one of the following commands at the prompt:

Windows — `CA.bat -newca -days 200` *or*

UNIX — `CA.sh -newca -days 200`

The expiration date of the generated certificate is specified in the openssl.cnf file. If the expiration date of the certificate does not match the certificate authority you specified using the `-days` option, OpenDeploy will assign the shorter expiration date of to certificates generated from the authority. See "Certificate Expiration" on page 303 for more information.

Generating a Certificate

Creating a certificate is very similar to creating the certificate authority and includes many of the same prompts for information. When generating a certificate, the authority is assumed to exist already. If you have one OpenDeploy sender and one OpenDeploy receiver, you must generate two certificates, one for the source host and one for the target host. You can also generate one certificate set and rename this set to be source and target keys. You must have a certificate key/pair for every OpenDeploy host you want to be included in SSL deployments.

To generate a certificate for an OpenDeploy host, follow these steps:

1. Navigate to the `od-home/bin` directory.
2. Generate a new certificate and key by entering the following command (depending on the platform) at the prompt:
   - Windows — **`CA.bat -certall`** *or*
   - UNIX — **`CA.sh -certall`**

   The certificate wrapper script generates RSA certificates only. To generate DSA certificates, do not use the CA scripts. Consult the OpenSSL Web site for more information.

3. Enter the appropriate information in response to the following prompt:

```
You are about to be asked to enter information that will be incorporated into
your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value, If you enter '.', the field
will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [California]:
Locality Name (eg, city) [Sunnyvale]:
Organization Name (eg, company) []:Interwoven
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, YOUR name) []:Receiver certificate
Email Address []:
```

   You cannot have two or more certificates with the exact same information. Each certificate must be unique. The difference between the certificate and the certificate authority can be identified by the different Common Name values. This value can be a reminder of the use to which you expect to put the certificate, for example, a receiver.

4. Answer yes at the prompts to sign and then commit the certificate:

```
Sign the certificate? [y/n]: y
```

```
1 out of 1 certificate requests certified, commit? [y/n]: y
```

   At the conclusion of these steps a private key file called `newreq.pem` and a certificate file called `newcert.pem` are generated.

5. Copy the generated certificate and key to the appropriate locations. A recommended place to store certificates and keys is:

   od-home/`cert`

   You must create this directory manually, as it is not generated during the installation. You should also rename the certificate and key to reflect their roles in the deployment cycle because newly generated certificate/key pairs will overwrite the previously existing `newreq.pem` and `newcert.pem` files. For example, name your source key files `odsrckey.pem` and `odsrccert.pem`, and your target key files `odtgtkey.pem` and `odtgtcert.pem`.

6. Remotely copy the generated certificates, private keys, and the CA certificate to the appropriate location on each peer host, depending on which peer host the certificate/key pair is intended. All OpenDeploy hosts using SSL encryption must have these items.

   Secure file transfer protocol (SFTP) and secure copy (SCP2) are good methods for moving these items to remote locations. For maximum security, you should physically transport them on a tape or diskette. Since you also have to move the certificates to the target host, you might choose to compress and package these items together into a `.tar` or `.zip` file before you do the transfers to peer hosts.

Certificate Expiration

The life span of the generated certificate is specified by the `default_days` attribute in the `openssl.cnf` file. This number is the expiration days for certificates `newcert.pem` generated based on `cacert.pem`. The default expiration period is 365 days.

If the expiration date of the certificate does not match the certificate authority you specified using the `-days` option, OpenDeploy assigns the shorter expiration date to certificates generated from the authority. In some cases, you might want to set the expiration date for certificate authority for a longer period and periodically expire the certificate using the same certificate authority.

Changing OpenSSL Defaults

Much of the process for generating certificates has been automated, and much of the inputs to the automation can be defaulted. These defaults are specified in the following configuration file:

```
od-home/bin/openssl.cnf
```

For example, should you need to relocate the `.rnd` file, you can modify the `RANDFILE` parameter in `openssl.cnf` to point to a different location before creating the certificate authority. OpenDeploy includes a configuration file for creating simple certificates. See “Obtaining Additional SSL Information” on page 296 for more information on modifying `openssl.cnf`.

SSL Configuration and Deployment Errors

Errors can occur when creating certificates or in setting up the deployments. Here are two of the most commonly reported error messages:

`PRNG not seeded` *and*

`Unable to write 'random state'`

These indicate that you have not seeded the random number generator with enough data. See “Setting Up the Certificate Authority” on page 298 for more information regarding seeding the random number generator.

If the following error message is displayed:

```
Self-signed certificate
```

this indicates that both the certificate and the certificate authority have the same name. You will discover this error when you try to use the two certificates together. Although you cannot generate two certificates with the same Distinguished Name, there is nothing to prevent you from generating the certificate authority and a certificate with the same name. The Distinguished Names of the two certificates must differ in at least one entry while creating the certificates.

You can look at the certificate generated from running the script with the `-certall` option, and observe the Distinguished Name of the issuing certificate authority. You can then regenerate the certificate and ensure that you are not reusing all of the certificate authority's name.See “Obtaining Additional SSL Information” on page 296 for more information regarding errors.

Verifying Certificates

You can verify the validity of the certificates you generate by entering the following command at the prompt:

Windows — `CA.bat -verify` *or*

UNIX — `CA.sh -verify`

If you have changed the name of the certificates since they were created, you must also add the certificate name to the command, for example:

```
CA.bat –verify odsrccert.pem or

CA.sh –verify odtgtcert.pem
```

If the certificate is valid, the following message is displayed:

```
certificate_name.pem: OK
```

For example, if you entered the following:

```
CA.bat -verify newcert.pem
```

the following message is displayed:

```
newcert.pem: OK
```

However, if there is a problem, OpenDeploy will display an error message such as the following:

```
newcert.pem:
/C=US/ST=California/L=Sunnyvale/O=Interwoven/OU=Engineering/CN=Engineering
Certificate Authority error 18 at 0 depth lookup:self signed certificate

/C=US/ST=California/L=Sunnyvale/O=Interwoven/OU=Engineering/CN=Engineering
Certificate Authority error 7 at 0 depth lookup:certificate signature failure
```

An error message like this can be the result of not using a unique Common Name when generating certificates. See “Generating a Certificate” on page 301 for more information.

Configuring OpenDeploy for SSL Data Transfer Encryption

After you generate and sign the certificates as described in the preceding sections, you must configure OpenDeploy to use SSL data transfer encryption. You can configure encryption using the following methods:

- In the base server and receiver configuration files.
- In the deployment configuration files.

Encryption values are specified in the `localNode` element of the base server or receiver configuration files of the OpenDeploy host. If you specify SSL data transfer encryption in these configuration files, then all incoming deployments are expected to be encrypted this way.

Encryption values in the deployment configuration are also specified in the `localNode` element, and the same attributes apply. Encryption in the deployment files only apply to that deployment.

In both cases, you must have the following attributes and their values specified within the `localNode` element:

- `sslCertificate` — enter the absolute path to the SSL public key certificate.
- `sslPrivateKey` — enter the absolute path to the SSL private key certificate.
- `sslCaCertificate` — enter the absolute path to the certificate authority. This allows OpenDeploy to authenticate the source from which the public and private key pairs for the source and target hosts are derived.
- `sslVerifyPeer` — (optional) indicate which of the following conditions apply in regards to the verification that the certificate authority for each public and private key pairs comes from the same source. This source is the value specified in the `sslCaCertificate` attribute.
  - `none` — no verification is performed. This is the default value.
  - `request` — verification is performed if the certificate/key pair exists on the peer of the host making the authentication request before the deployment can occur.
  - `require` — verification must be performed, and the certificate/key pair must exist on the peer of the host making the request before the deployment can occur.

Here is an example of how the `localNode` element and its encryption-related attributes might be configured for SSL data transfer encryption in the base server configuration file or in a deployment configuration:

```
<localNode
   host="mars"
   sslCertificate="od-home/cert/odsrccert.pem"
   sslPrivateKey="od-home/cert/odsrckey.pem"
   sslCaCertificate="od-home/bin/demoCA/certs/cacert.pem"
   sslVerifyPeer="request"
   />
```

Here is an example of how the `localNode` element and its encryption-related attributes might be configured for SSL data transfer encryption in the target receiver configuration file:

```
<localNode
   host="venus"
   sslCertificate="od-home/cert/odtgtcert.pem"
   sslPrivateKey="od-home/cert/odtgtkey.pem"
   sslCaCertificate="od-home/bin/demoCA/certs/cacert.pem"
   sslVerifyPeer="request"
   />
```

Ciphers

You can specify various ciphers to use in encryption. During a connection, the OpenDeploy source and target hosts will negotiate over which cipher to use. During the negotiation phase, OpenDeploy selects the highest priority cipher that both source and target hosts support.The use of ciphers is specified by the presence of the `sslCiphers` attribute in the `localNode` element, located in the base server or receiver configuration file. For example:

```
sslCiphers="cipherlist"
```

where *`cipherlist`* contains one or more ciphers, ranked left to right from highest priority to lowest priority, separated by colons (“:”). For example:

```
sslCiphers="EDH-RSA-DES-CBC3-SHA:DES-CBC-SHA"
```

The following is a list of all cipher strings and their meanings:

- `DH` — cipher suites using `DH`, including anonymous `DH`.

- ADH — anonymous DH cipher suites.
- 3DES — cipher suites using triple DES.
- DES — cipher suites using DES (not triple DES).
- RC4 — cipher suites using RC4.
- RC2 — cipher suites using RC2.
- IDEA — cipher suites using IDEA.
- MD5 — cipher suites using MD5.
- SHA1, SHA — cipher suites using SHA1.

Refer to the following Web site for more information on ciphers:

```
www.openssl.org/docs/apps/ciphers.html
```

OpenDeploy allows you to use the following types of ciphers:

- No-authentication ciphers:
  - None
- Low-strength (56 and 40 bit) ciphers:
  - EXP1024-RC4-SHA (56 bit)
  - EXP1024-DES-CBC-SHA (56 bit)
  - EXP1024-RC2-CBC-MD5 (56 bit)
  - EXP1024-RC4-MD5 (56 bit)
  - EDH-RSA-DES-CBC-SHA (56 bit)
  - DES-CBC-SHA (56 bit)
  - EXP-EDH-RSA-DES-CBC-SHA (40 bit)
- Medium-strength (128 bit) ciphers:
  - – RC4-SHA
  - – RC4-MD5 SSLversion 2 or version 3

- High-strength (168 bit) ciphers:
  - EDH-RSA-DES-CBC3-SHA
  - DES-CBC3-SHA

If `sslCiphers` is not specified, OpenDeploy tries each supported cipher, starting from the high-strength ones, until a compatible cipher is found with the remote OpenDeploy host. If no compatible is found, the deployment will fail.

The following example adds a 168-bit cipher and a low-strength cipher to the SSL data transfer key encryption code created in “Configuring OpenDeploy for SSL Data Transfer Encryption” on page 306:

```
<localNode
   host="mars"
   sslCertificate="od-home/cert/odsrccert.pem"
   sslPrivateKey="od-home/cert/odsrckey.pem"
   sslCaCertificate="od-home/bin/demoCA/certs/cacert.pem"
   sslVerifyPeer="request"
   sslCiphers="EDH-RSA-DES-CBC3-SHA:DES-CBC-SHA"
   />
```

Testing the SSL Encryption Configuration

After you have configured OpenDeploy for SSL encryption, you should test it before performing an actual deployment. This section instructs you to modify the sample deployment configuration file `test.xml`. This sample deployment directs your OpenDeploy base server to deploy files to itself. Therefore, for this test, you will configure your base server configuration file to receive. However, you can also modify the `test.xml` file to deploy to other hosts instead of, or in addition to, the sending host.

To test your SSL encryption configuration, follow these steps:

1. Navigate to the following location:

   ```
   od-home/conf
   ```

2. Make a copy of the file `test.xml` and rename it `testssl.xml`.

3. Modify the `localNode` element in the `testssl.xml` file as follows:

```
<localNode
    host="host_name"
    sslCertificate="od-home/cert/odsrccert.pem"
    sslPrivateKey="od-home/cert/odsrckey.pem"
    sslCaCertificate="od-home/bin/demoCA/certs/cacert.pem"
    sslVerifyPeer="request"
    sslCiphers="EDH-RSA-DES-CBC3-SHA:DES-CBC-SHA"
    />
```

4. Navigate to the following location:

   od-home`/etc`

5. Modify the `localNode` element in the base server configuration file (by default `odbase.xml`) as follows:

```
<localNode
    host="host_name"
    sslCertificate="od-home/cert/odtgtcert.pem"
    sslPrivateKey="od-home/cert/odtgtkey.pem"
    sslCaCertificate="od-home/bin/demoCA/certs/cacert.pem"
    sslVerifyPeer="request"
    sslCiphers="EDH-RSA-DES-CBC3-SHA:DES-CBC-SHA"
    />
```

   If `testssl.xml` has been configured to deploy to other hosts as well, each target's host server configuration file must be modified in this way.

6. Stop and restart the OpenDeploy service or daemon on each host whose base server or receiver configuration file was modified in the previous step. See "Stopping OpenDeploy" on page 53 and "Starting OpenDeploy" on page 47 for more information.

7. Run the `testssl.xml` deployment.

If the deployment runs successfully, the SSL encryption is correctly set up. If the deployment fails, or if the base server software does not appear to have started properly, refer to the OpenDeploy base server and deployment log files to determine and correct the problem. See Chapter 6, "Logging" on page 243 for more information.

Logging

You can verify that a deployment was run with SSL encryption by checking the receiver micro deployment log file. An entry indicating that SSL encryption was used in the deployment is written immediately below the date time stamp. For example:

```
v========== Start Log [Mon Jun 03 15:18:48 2002] ==========
(2) Using SecureSocketsLayer protocol
```

See “Receiver Micro Deployment Log” on page 256 for more information on this type of log file.

Chapter 9

Composing Deployments

This chapter describes the Deployment Configuration Composer, and how to use it to create complete functional deployment configurations through the OpenDeploy user interface. Although the ability to write and understand XML code is not required to create and edit deployment configurations using the Deployment Configuration Composer, it is recommended that you review the features and functionality described in Chapter 3, "Deployment Types" and Chapter 4, "Deployment Features" prior to creating and editing any new or existing deployment configurations.

Deployment Configuration Composer

The Deployment Configuration Composer is a tool within the OpenDeploy user interface for creating new deployment configurations and editing existing ones. You can author or edit the configuration of any deployment that resides in the *od-admin*/`conf` directory, and that conforms to XML-based structure used in OpenDeploy 5.x deployment configurations.

The text boxes, lists, and option buttons that appear in the Deployment Configuration Composer interface correspond to elements and attributes in the deployment configuration file. The values you input are added to the deployment configuration file. Any existing deployment configuration can be edited and updated using the Deployment Configuration Composer.

To access the Deployment Configuration Composer, follow these steps:

1. Select **Configurations > View Configurations** to display the Deployment Configuration window.
2. Select the OpenDeploy host on which the deployment configuration will reside from the **Selected Host** list.
3. Click **New** to open a new browser window containing the containing the Deployment Configuration Composer (Figure 63).

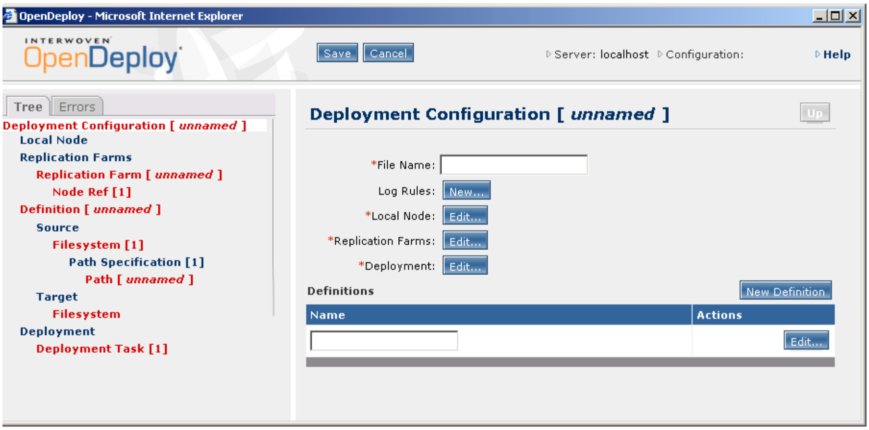
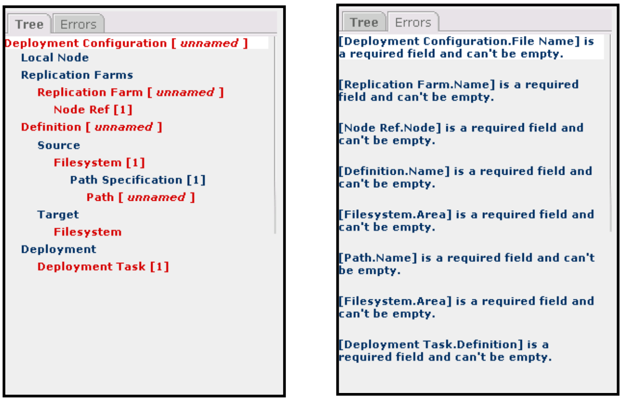


Figure 63: Deployment Configuration Composer

To edit an existing deployment configuration, select that configuration from the **Deployment** list and click **Edit** to open the Deployment Configuration Composer displaying the deployment configuration you selected.

Tree and Errors Tabs

The Deployment Configuration Composer contains Tree and Errors tabs on the left that display either a tree of configuration elements from which you can select, or a list of errors that display omissions of required information in your deployment configuration (Figure 64).



- Deployment Configuration [*unnamed*]
  - Local Node
  - Replication Farms
    - Replication Farm [*unnamed*]
      - Node Ref [1]
  - Definition [*unnamed*]
    - Source
      - Filesystem [1]
        - Path Specification [1]
          - Path [*unnamed*]
    - Target
      - Filesystem
  - Deployment
    - Deployment Task [1]

[Deployment Configuration.File Name] is a required field and can't be empty.

[Replication Farm.Name] is a required field and can't be empty.

[Node Ref.Node] is a required field and can't be empty.

[Definition.Name] is a required field and can't be empty.

[Filesystem.Area] is a required field and can't be empty.

[Path.Name] is a required field and can't be empty.

[Filesystem.Area] is a required field and can't be empty.

[Deployment Task.Definition] is a required field and can't be empty.

Figure 64: Tree and Errors Tabs

Tree Tab

The Tree tab displays the required and optional elements that make up the deployment configuration. Those elements in red are required to be completed before the configuration is done. After the information corresponding to a red element entry has been provided, the entry will turn blue.

Values contained in brackets ([ ]) reflect either a unique name value you assign them, for example:

```
Deployment Configuration [MYCONFIGURATION]
```

or the numbered occurrence of that elements, for example:

```
Filesystem [2]
```

When you complete adding information to a window in the Details pane, you can display the previous window by clicking the **Up** button. You can also display another window by selecting its entry in the Tree pane. When you provide names for elements, such as the definition element, that name is displayed next to its element in the Tree tab.

Errors Tab

The Errors tab displays error messages associated with any elements, attributes, or values that are missing or incorrect. You then must complete the required information and save the file again. When you save a deployment configuration, the Errors tab will automatically display any associated errors.

Details Pane

Selecting an entry in the Tree tab displays the corresponding window in the Details pane on the right of the browser window. Each window contains text boxes, buttons, drop-down lists, and other features which correspond to attributes in the deployment configuration. In some cases you must enter values for these attributes. In other cases, you have the option of providing a value, or allowing OpenDeploy to use its default values. Those attributes with a red asterisk (*) to the left denote required attributes for you to complete. In some cases, you can access a window in the Details pane either by selecting an entry in the tree, or by clicking a button in another Details pane window.

Types of Deployment Configuration Settings

Deployment configuration settings fall under the following categories:

- Global settings — configuration applies to the deployment as a whole. These settings include logging, nodes and node farms, transactional capability, and Deploy and Run scripting.
- Definition settings — configuration applies to the specified definition element, which consists of a source/target pairing and its associated rules. These settings include the source and target areas, filters, and deployment rules.

Global Deployment Settings

Global deployment settings apply to all sources, targets, and deployed files included in the deployment. The following required tasks apply:

- Name the deployment.
- Verify or change the source host name.
- Name your default node farm element.

- Assign one or more target hosts listed in your nodes configuration file to each node farm.
- Indicate whether or not the deployment is transactional.

You can apply perform these optional tasks:

- Set your log rules.
- Configure encryption.
- Add and name any additional node farm elements.
- Enter the name of the deployment that your target host will run after receiving your deployed files. Also indicate whether the target host will invoke a new deployment if your deployment is not successful. These tasks are required for multi-tiered deployments only.
- Assign Deploy and Run capabilities as necessary, including those scripts that trigger upon the deployment of particular files, directories, or deployments.

Definition Settings

A deployment configuration contains one or more definition elements. Each definition element contains additional elements and attribute values that identify the source locations where deployed files originate, and the target location where those deployed files are received. For comparison-based deployments, those file locations participating in the deployment also are specified. Any rules establishing the deployment eligibility criteria of the files are also contained within the definition.

The following required tasks apply to each instance of the definition element in your deployment configuration:

- Name the default definition element.
- Indicate whether the definition is a forward or reverse deployment.
- Specify the type of location (file system or TeamSite area) where the files to be deployed reside on the source host.
- Enter the path to the source file location, including previous TeamSite area or file list path.

- Enter any subdirectories within the specified source file location that further define where the source files reside. If you do not wish to further refine the area specification, you must enter the value "." to indicate that no subdirectory is specified.
- Enter the path to the target file location.

You can also apply these optional tasks:

- Add and name any additional definition elements.
- Add and configure any file path or pattern exclusion rules, and whether or not to follow symbolic links during the deployment.
- Configure any target area overrides for deployments with multiple source area locations. Apply any transfer, comparison, or permission rules to those files deployed to this alternate area.
- Add and configure any additional source file locations.
- Configure any comparison, transfer, or permission rules for the deployment.
- Configure any source- or target-side filters.

Creating a New Configuration

The following section leads you through the process of creating a new deployment configuration using the Deployment Configuration Composer. Each major task is separated to make the procedure easier to learn.

To create a new deployment configuration using the Deployment Configuration Composer, follow these steps:

1. Select **Configurations > View Configurations** to display the Deployment Configuration window.
2. Select the OpenDeploy host on which your new deployment configuration will reside from the **Selected Host** list.
3. Click **New**. A new browser opens containing the Deployment Configuration Composer. The Deployment Configuration window (Figure 65) is displayed within the Deployment Configuration Composer. Here is where you will begin creating your new deployment configuration.

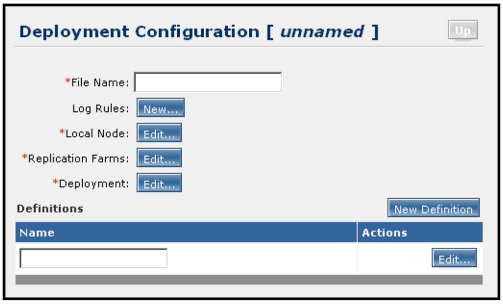


Figure 65: Deployment Configuration Window

Naming the Deployment Configuration

Enter the file name of the deployment configuration in the **File Name** box.

You can enter a file name of any length up to the limit supported by the host operating system. Deployment configurations residing on UNIX hosts cannot have spaces in the file names. Those running on Windows hosts may contain spaces. It is not necessary to include the `.xml` extension in the file name you enter. The Configuration Composer will automatically add that extension when you save the file.

Specifying the Log Rules

The log rules in a deployment configuration determine the maximum size a log file can grow before that file is closed to new entries, and a new log file is started. This is known as the rollover threshold. You can also specify whether you want the logging levels to be normal or verbose. See Chapter 6, "Logging" on page 243 for a complete description on OpenDeploy logging.

The default settings for Log Rules comes from the base server and receiver configuration files, however you can provide specific values for your deployment by clicking the **New** button associated with the **Log Rules** to display the Log Rules window (Figure 66).

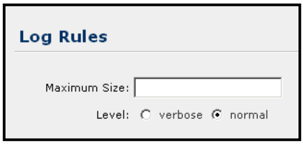


Figure 66: Log Rules Window

You can also select the **Log Rules** entry in the Tree. Here you can set your own maximum log file size and logging level. You can input the following values in the Log Rules window:

- **Maximum Size** box — enter the rollover threshold value. You can specify different byte measurements in the value, including megabytes (`mb`), kilobytes (`kb`), and bytes (`b`). For example:

  `10mb` *or*

  `10000kb` *or*

  `10000000b`

  If you enter a numerical value to fail to provide the measurement indicator, OpenDeploy will default to bytes (`b`). However, the minimum allowable size is 100 KB, or 102400 bytes. Setting a value smaller than this will be overridden with the default minimum when the deployment is run. See "Logging Configuration Settings" on page 258 for more information on OpenDeploy default logging values and rules.

- **Level** options — select the level and type of logging OpenDeploy will perform:
  - **verbose** — logs a high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
  - **normal** — logs standard status and error messages. In most cases, this level of logging provides sufficient detail to meet your needs.

Click **Up** when you are done with the Log Rules window to return to the Deployment Configuration window.

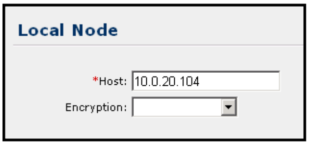
Specifying logging rules in the Deployment Configuration Composer is optional. If no logging levels are specified, OpenDeploy will use the logging levels present in the base server configuration if present, or use the following default settings:

- **Maximum Size** (rollover threshold): 32 MB
- **Level**: Verbose

Refer to "Logging" on page 77 in the *OpenDeploy Installation and Reference Guide* for more information on setting base server logging.

Verifying or Changing the Source Host Name

Your deployment configuration must include the name of your OpenDeploy host server. This name can either be its fully qualified DNS server name or its IP address. Click the **Edit** button associated with the **Local Host** in the Deployment Configuration window to display the Local Node window (Figure 67).



Local Node

*Host: 10.0.20.104

Encryption:

Figure 67: Local Node Window

By default, your host's name will be displayed in the **Host** box. However, you can change that value to another OpenDeploy host if you want. An example of this would be if you are authoring a deployment configuration you intend to use on another OpenDeploy host.

Specifying Deployment Encryption

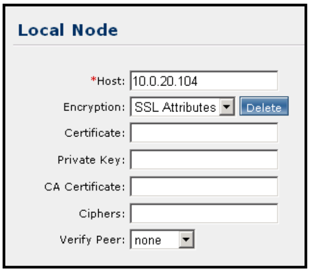
Also present in the Local Node window (Figure 67) is your deployment encryption settings. If you want to use encryption with your deployment configuration, you must specify it here in the **Encryption** list.

- **SSL Attributes** — symmetric key encryption that uses the secure sockets layer (SSL) key certificate.
- **Key Fil**e — 40-bit symmetric encryption that uses a key file.

If you select an encryption option, the following additional configuration attributes for that option will be displayed in the window for you to complete. See Chapter 8, "Encryption" on page 293 for more information on how encryption in OpenDeploy works.

SSL Attributes

The following attributes are displayed when you select the **SSL Attributes** encryption option (Figure 68):



Local Node

*Host: 10.0.20.104

Encryption: SSL Attributes

Certificate:

Private Key:

CA Certificate:

Ciphers:

Verify Peer: none

Figure 68: SSL Attributes

- **Certificate** box — enter the absolute path to the SSL public key certificate. This attribute is required for using asymmetric key encryption.

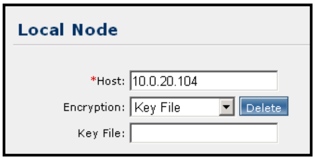
- **Private Key** box — enter the absolute path to the SSL private key certificate. This attribute is required for using asymmetric key encryption.
- **CA Certificate** box — enter the absolute path to the certificate authority. This allows OpenDeploy to authenticate the source from which the public and private key pairs for the source and target hosts are derived. This attribute is required for using asymmetric key encryption.
- **Ciphers** box — enter the SSL ciphers to use. Multiple ciphers must be separated by colons (":"). For example:

  ```
  EDH-DSS-DES-CBC3-SHA:EXP-EDH-DSS-DES-CBC-SHA
  ```

  This attribute is optional. Use of asymmetric encryption does require the use of ciphers.
- **Verify Peer** list — select which of the following conditions apply in regards to the verification that the certificate authority for each public and private key pairs comes from the same source. This source is the value specified in the **sslCaCertificate** attribute.
  - **none** — no verification is performed. This is the default value.
  - **request** — verification is performed if the certificate/key pair exists on the peer of the host making the authentication request before the deployment can occur.
  - **require** — verification must be performed, and the certificate/key pair must exist on the peer of the host making the request before the deployment can occur.

Key File

The following attributes are displayed when you select the **Key File** encryption option (Figure 69):



Local Node

*Host: 10.0.20.104

Encryption: Key File

Key File:

Figure 69: Key File Attributes

- **Key File** box— specifies the absolute path to the key file that provides the weak 40-bit symmetric encryption. For example:

  `C:\secure\MyKeyFile.txt` *or*

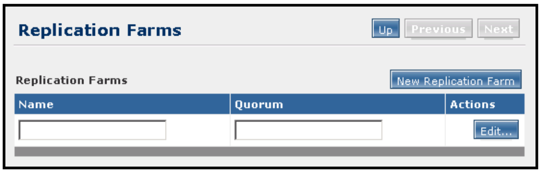
  `/secure/MyKeyFile.txt`

Click **Up** when you are done with the Local Node window to return to the Deployment Configuration window.

Naming the Replication Farm Element

The replication farm is an element in the deployment configuration that represents a collection of one or more target hosts.You must give each replication farm a unique name.

Click the **Edit** button associated with the **Replication Farms** to display the Replication Farms window (Figure 70).



Name	Quorum	Actions
		Edit...

Figure 70: Replication Farms Window

Here you must enter a unique name for your replication farm set in the **Name** box, for example:

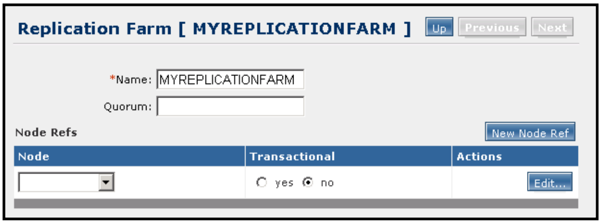
```
MYREPLICATIONFARM
```

If you want more than one replication farm in your deployment configuration, click **New Replication Farm** to add another **Replication Farms** entry. You must provide a unique name for each replication farm in your deployment configuration.

You can assign a quorum value for each replication farm in the associated **Quorum** box. The quorum value specifies the number of target hosts that must successfully receive their deployments for the overall deployment to be considered successful. A transactional fan-out deployment whose quorum value was not met would be rolled back, even if some of the targets received their deployments successfully.

Adding Target Host Nodes to the Replication Farm Element

After you have named your replication farm elements, you must assign individual target host nodes to each one. Click the **Edit** button associated with a farm entry in the Replication Farms window to display the Replication Farm window for that particular replication farm(Figure 71).



Replication Farm [MYREPLICATIONFARM]

*Name: MYREPLICATIONFARM

Quorum:

Node Refs

Node	Transactional	Actions
	○ yes ● no	Edit...

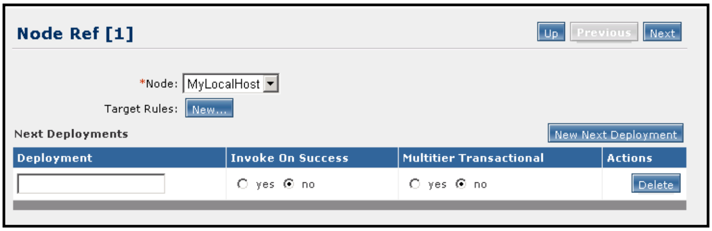
Figure 71: Replication Farm Window

Here you can select those target hosts listed in your nodes configuration file (by default `odnodes.xml`) from the **Node** list. If you want to assign more than one target node to the replication farm element, click **New Node Ref** to add another entry.

Assigning Next-Tier Deployments to Target Hosts

If you intend for one or more of your deployment target hosts to redeploy the files it receives, you can configure that target host to trigger a specific deployment of its own upon receipt of the initial deployed files. This is part of a next-tier deployment. Any target you assign a next-tier deployment must have the base server software installed, and be able to deploy files to targets of its own.

Click the **Edit** button associated with your node entry in the Farm window to display the Node Ref window (Figure 72).



Node Ref [1]

*Node: MyLocalHost

Target Rules: New...

Next Deployments

Deployment	Invoke On Success	Multitier Transactional	Actions
	○ yes ● no	○ yes ● no	Delete

Figure 72: Node Ref Window

Enter the name of the deployment you want the associated node to run following receipt of the initial deployment in the **Deployment** box. The deployment you enter must be present in the target host.

If you select the **yes** option under **Invoke On Success**, then the next-tier deployment will only be run if the target host receives the first deployment successfully.

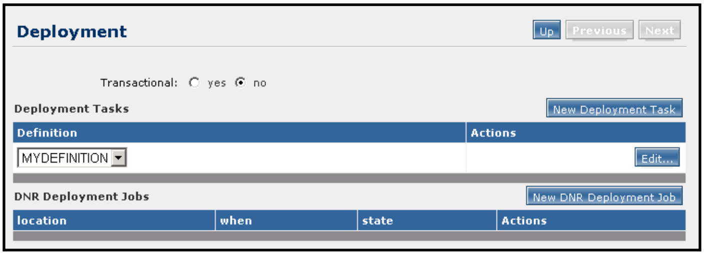
If the target is part of a multi-tiered deployment, and you want its files restored to its previous state in the event the multi-tiered deployment is unsuccessful, select the **yes** option under **Multitier Transactional**.

Click **New Next Deployment** if you want to assign another next-tier deployment to your target host. You can also select a different target host from the **Node** list to which you can associate your next-tier deployment.

Specifying a Transactional Deployment.

A transactional deployment will automatically roll back a deployment and restore the target host's files to their previous states in the event the deployment is not completely successful. This feature is particularly useful if you are deploying to multiple targets simultaneously, and it is important that all the target hosts contain the exact same files. See "Transactional Deployments" on page 144 for more information.

Select **Deployment** from the tree to display the Deployment window (Figure 73).



Transactional: ○ yes ● no

Deployment Tasks

Definition	Actions
MYDEFINITION	Edit...

DNR Deployment Jobs

location	when	state	Actions

Figure 73: Deployment Window

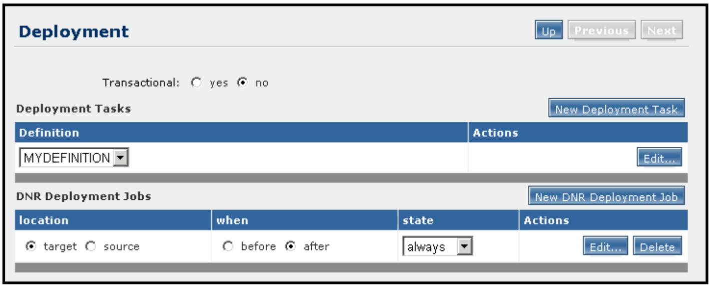
To make your deployment transactional, click the **yes** button associated with the **Transactional** feature in the Deployment window.

Setting Deploy and Run

Deploy and Run is an OpenDeploy feature that allows you to run an external script during a deployment when one or more triggers apply. These triggers can include when a certain individual or category of directories or files are deployed, or when a particular deployment itself is run. Deploy and Run scripts associated with these triggers can be set to run on either the source host running the deployment, the host receiving the deployed files. They can also be set to trigger on the success of a deployment, its failure, or both. See "Deploy and Run" on page 201 for more information on this feature.

Assigning Deployment-based Deploy and Runs

You can assign a deployment-based Deploy and Run to a deployment by clicking **New DNR Deployment Job** in the Deployment window. This results in additional attributes being displayed for configuring a Deploy and Run (Figure 74).



Transactional: ○ yes ● no

Deployment Tasks

Definition	Actions
MYDEFINITION ▾	Edit...

DNR Deployment Jobs

location	when	state	Actions
● target ○ source	○ before ● after	always ▾	Edit... Delete

Figure 74: Deployment Window with Deploy and Run Attributes Displayed

Select the **target** or **source** option under **location** to indicate on which end of the deployment the Deploy and Run will run.

Select the **before** or **after** option under **when** to indicate when during the deployment the Deploy and Run will run.

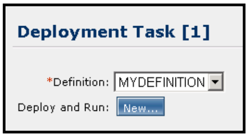
Select the appropriate option from the **state** list to indicate whether the Deploy and Run script should run as a result of the **success** or **failure** of the deployment, or whether it should **always** run in either case. If the **when** option is set to **before**, then the **state** attribute is implicitly set to **always**, because there has been no deployment yet to determine success or failure.

Click **Edit** to display the DNR Deployment Job window, where you can configure your deployment-based Deploy and Run. See "DNR Deployment" on page 334 for more information.

Click **Delete** to remove the associated Deploy and Run configuration.

Assigning Other Deploy and Runs Types

You can assign file- and directory-based Deploy and Runs, as well as deployment-based ones, in the Deployment Task window (Figure 75).



Deployment Task [1]

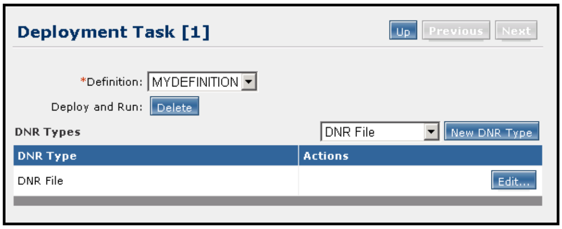
*Definition: MYDEFINITION

Deploy and Run:

Figure 75: Deployment Task Window

The Deployment Task window is accessible by selecting **Deployment Task** from the tree, or by clicking the **Edit** button associated with each definition entry in the Deployment window.

Select the definition element to which you want to associate the Deploy and Run by selecting its name from the **Definition** list. Click **New** to display additional Deploy and Run attributes in the Deployment Task window (Figure 76).



Deployment Task [1]

*Definition: MYDEFINITION

Deploy and Run:

DNR Types: DNR File

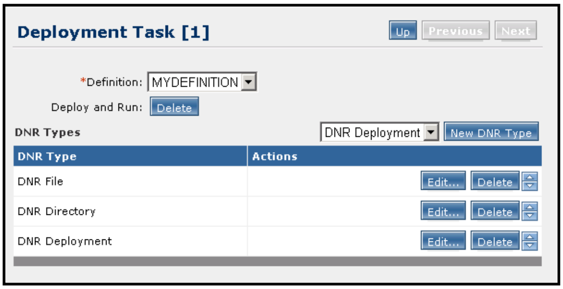
DNR Type	Actions
DNR File	Edit...

Figure 76: Deployment Task Window with Deploy and Run Attributes

The Deploy and Run feature includes the following options:

- **DNR File** — select to specify under what conditions deployed files can trigger a Deploy and Run script.
- **DNR Directory** — select to specify under what conditions deployed directories can trigger a Deploy and Run script.
- **DNR Deployment** — select to specify under what conditions the running of a particular deployment can trigger a Deploy and Run script.

Select the appropriate choice from the **Type** list and click **New DNR Type** to add an entry for that Deploy and Run element. You can create elements for DNR File, DNR Directory, DNR Deployment, in any number and combination (Figure 77).



Deployment Task [1]

*Definition: MYDEFINITION

Deploy and Run:

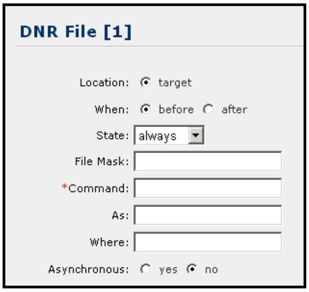
DNR Types: DNR Deployment

DNR Type	Actions
DNR File	Edit... Delete
DNR Directory	Edit... Delete
DNR Deployment	Edit... Delete

Figure 77: Deployment Task Window with Multiple Deploy and Run Elements

DNR File

Clicking the **Edit** button associated with a DNR File entry displays the DNR File window (Figure 78).



DNR File [1]

Location: ● target

When: ● before ○ after

State: always

File Mask:

*Command:

As:

Where:

Asynchronous: ○ yes ● no

Figure 78: DNR File Window

The DNR File window contains the following attributes:

- **Location** option — this option is fixed as **target**, indicating the script will take place on the target host.

- **When** option — select whether the script should be executed **before** or **after** the deployment of the particular file.

- **State** list — indicates whether the Deploy and Run script should run as a result of the **success** or **failure** of the deployment, or whether it should **always** run in either case.

  If the **When** option is set to **before**, then the **State** attribute is implicitly set to **always**, because there has been no deployment yet to determine success or failure.

- **File Mask** box — enter the regular expression specifying the deployed files that will trigger the script. If you entered the following value:

  ```
  \.html$
  ```

  any deployed file with the file extension `.html` will trigger the Deploy and Run script.

- **Command** box — enter the command where OpenDeploy can start a Deploy and Run script, as well as any accompanying flags or options. You can also specify an executable invocation line. For example:

  `C:\bin\email_to_admin.bat -user jdoe@interwoven.com` *or*

  `/bin/mail jdoe@interwoven.com < /tmp/message.txt`

  If the command you are going to run requires a scripting engine, the scripting engine must be on the PATH of the user (or system, on Windows) who will be running the script or specified with a full path). For example:

  `/bin/sh /usr/local/bin/email_to_admin.sh -u jdoe@interwoven.com` *or*

  `/usr/local/bin/iwperl /path/to/script.pl`

- **As** box — enter a different user name or user ID under which you can run the script. If you entered the following value:

  `rroe` *or*

  `100`

  you could run the script as rroe rather than as your regular user name. By default, the script runs as the user who invokes OpenDeploy, who will need to be root for most purposes.

  This feature applies to UNIX hosts only. On Windows hosts, this feature always runs as *Default User*.

- **Where** box — enter the path to the location where the **Command** attribute value is to be run. For example:

  `/tmp` *or*

  `C:\temp`

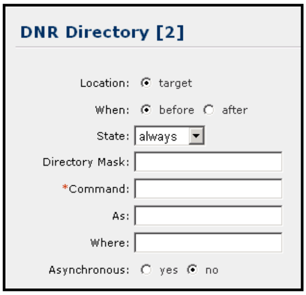
  You must use the path syntax supported by your OpenDeploy server. Forward slashes (“/”) can be used for either Windows or UNIX hosts, while backslashes (“\”) are only permitted on Windows hosts.

  The **Where** attribute is optional. If you do not specify a value, the process takes place in the root directory.

- **Asynchronous** option — select **yes** to run the script asynchronously or **no** not to. Exercise caution when using this mode, as it could cause many scripts to be run simultaneously. The output from scripts run asynchronously is not captured.

DNR Directory

Clicking the **Edit** button associated with a DNR Directory entry displays the DNR Directory window (Figure 79).



DNR Directory [2]

Location: ● target

When: ● before ○ after

State: always

Directory Mask:

*Command:

As:

Where:

Asynchronous: ○ yes ● no

Figure 79: DNR Directory Window

The DNR Directory window contains the following attributes:

- **Location** option — this option is fixed as **target**, indicating the script will take place on the target host.
- **When** option — select whether the script should be executed **before** or **after** the deployment of the particular directory.
- **State** list — select whether the Deploy and Run script should run as a result of the **success** or **failure** of the deployment, or whether it should **always** run in either case.

  If the **When** option is set to **before**, then the **State** attribute is implicitly set to **always**, because there has been no deployment yet to determine success or failure.

- **Directory Mask** box — enter the regular expression specifying the deployed directories that will trigger the script. If you entered the following value:

  ```
  cgi-bin$
  ```

  any deployed directory in the deployment path named cgi-bin will trigger the Deploy and Run script.

- **Command** box — enter the command where OpenDeploy can start a Deploy and Run script, as well as any accompanying flags or options. You can also specify an executable invocation line. For example:

  `C:\bin\email_to_admin.bat -user jdoe@interwoven.com` *or*

  ```
  /bin/mail jdoe@interwoven.com < /tmp/message.txt
  ```

  If the command you are going to run requires a scripting engine, the scripting engine must be on the PATH of the user (or system, on Windows) who will be running the script or specified with a full path). For example:

  `/bin/sh /usr/local/bin/email_to_admin.sh -u jdoe@interwoven.com` *or*

  ```
  /usr/local/bin/iwperl /path/to/script.pl
  ```

- **As** box — enter a different user name or user ID under which you can run the script. If you entered the following value:

  `rroe` *or*

  ```
  100
  ```

  you could run the script as rroe rather than as your regular user name. By default, the script runs as the user who invokes OpenDeploy, who will need to be root for most purposes.

  This feature applies to UNIX hosts only. On Windows hosts, this feature always runs as *Default User*.

- **Where** box — enter the path to the location where the **Command** attribute value is to be run. For example:

  `/tmp` *or*

  `C:\temp`

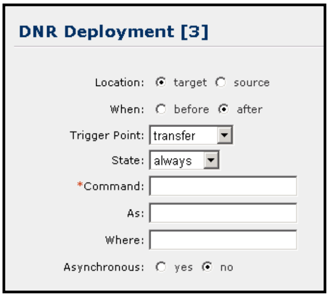
  You must use the path syntax supported by your OpenDeploy server. Forward slashes ("/") can be used for either Windows or UNIX hosts, while backslashes ("\") are only permitted on Windows hosts.

  The **Where** attribute is optional. If you do not specify a value, the process takes place in the root directory.

- **Asynchronous** option — select **yes** to run the script asynchronously or **no** not to. Exercise caution when using this mode, as it could cause many scripts to be run simultaneously. The output from scripts run asynchronously is not captured.

DNR Deployment

Clicking the **Edit** button associated with a DNR Deployment entry displays the DNR Deployment window (Figure 80).



DNR Deployment [3]

Location: ● target ○ source

When: ○ before ● after

Trigger Point: transfer

State: always

*Command:

As:

Where:

Asynchronous: ○ yes ● no

Figure 80: DNR Deployment Window

The DNR Deployment window contains the following attributes:

- **Location** option — select whether the Deploy and Run script is taking place on the **source** or **target** host.
- **When** option — select whether the script should be executed **before** or **after** the deployment of the particular file.
- **State** list — indicates whether the Deploy and Run script should run as a result of the **success** or **failure** of the deployment, or whether it should **always** run in either case.

  If the **When** option is set to **before**, then the **State** attribute is implicitly set to **always**, because there has been no deployment yet to determine success or failure.
- **Command** box — enter the command where OpenDeploy can start a Deploy and Run script, as well as any accompanying flags or options. You can also specify an executable invocation line. For example:

  `C:\bin\email_to_admin.bat -user jdoe@interwoven.com` *or*

  `/bin/mail jdoe@interwoven.com < /tmp/message.txt`

  If the command you are going to run requires a scripting engine, the scripting engine must be on the PATH of the user (or system, on Windows) who will be running the script or specified with a full path). For example:

  `/bin/sh /usr/local/bin/email_to_admin.sh -u jdoe@interwoven.com` *or*

  `/usr/local/bin/iwperl /path/to/script.pl`
- **As** box — enter a different user name or user ID under which you can run the script. If you entered the following value:

  `rroe` *or*

  `100`

  you could run the script as rroe rather than as your regular user name. By default, the script runs as the user who invokes OpenDeploy, who will need to be root for most purposes.

  This feature applies to UNIX hosts only. On Windows hosts, this feature always runs as *Default User*.

- **Where** box — enter the path to the location where the **Command** attribute value is to be run. For example:

  `/tmp` *or*

  `C:\temp`

  You must use the path syntax supported by your OpenDeploy server. Forward slashes ("/") can be used for either Windows or UNIX hosts, while backslashes ("\") are only permitted on Windows hosts.

  The **Where** attribute is optional. If you do not specify a value, the process takes place in the root directory.

- **Asynchronous** option — select **yes** to run the script asynchronously or **no** not to. Exercise caution when using this mode, as it could cause many scripts to be run simultaneously. The output from scripts run asynchronously is not captured.

Deleting Deploy and Run Entries

Each Deploy and Run element has associated attributes that you can display the complete by clicking the **Edit** button associated with the specific Deploy and Run element. If your deployment already has a Deploy and Run element configured, **a Delete** button will be displayed in place of the **New** button. Click this **Delete** button to delete all of the Deploy and Run elements you have created for the associated definition. If you want to delete specific Deploy and Run elements, but leave other intact, click the **Delete** button of the associated Deploy and Run element entry.

Naming and Adding Definitions

A definition element in a deployment configuration specifies each pairing of one or more file system locations or TeamSite areas residing on the source host with a single target location that will receive the deployed files. In the case of file system comparison deployments, the files residing in the target location will also be compared with those in the source host file system location. You can have more than one definition element in a deployment configuration, and each one must have a unique name.

Enter a unique name for your definition in the **Definition** box in the Deployment Configuration window (Figure 65), for example:

```
MYDEFINITION
```

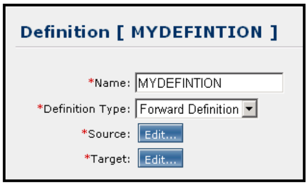
If you want more than one definition in your deployment configuration, click **New Definition** to add another **Definition** entry. You must provide a unique name for each definition in your configuration.

Selecting the Definition Type

Each definition within a deployment can be one of the following types:

- **Forward Definition** — the deployment originates at source host (usually a development server) and sends files to a receiving host (usually a production server). This type of deployment definition follows the standard development workflow.
- **Reverse Definition** — the deployment originates at a receiving host (usually a production server) and deploys files back to the source host (usually a development server). This type of deployment definition will resynchronize the files between a development server and a production server if the production server has its files modified outside the standard workflow. See "Reverse Deployments" on page 156 for more information

You can select the definition type in the Definition window (Figure 81).



Definition [MYDEFINTION]

*Name: MYDEFINTION

*Definition Type: Forward Definition

*Source: Edit...

*Target: Edit...

Figure 81: Definition Window

Select **Definition** from the tree to open this window. You can also click the **Edit** button associated with your definition entry in the Deployment Configuration window (Figure 73).

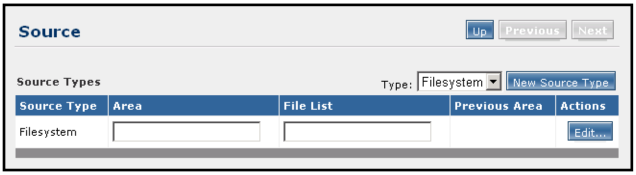
Select the type of definition you want from the **Definition Type** list.

Defining the Source File Location

The source file location is where the deployable files reside on the source host. You can configure your source file location in the Source window. Select **Source** from the tree to display the Source window. You also can click the **Edit** button associated with the source in the Definition window (Figure 81). The contents of the Source window is determined by the type of deployment you are configuring.

Directory Comparison Deployments

For directory comparison deployments, the Source window (Figure 82) displays attributes for the source file location and for overriding the target file location. The file list attribute only applies to file list deployments described later in this section. This is the default Source window displayed when you first access the Source window.



Source

Source Types

Type: Filesystem

Source Type	Area	File List	Previous Area	Actions
Filesystem				Edit...

Figure 82: Source Window for Directory Comparison Deployments

Enter the absolute path to the file system location where your deployment's source files reside in the **Area** box. For example:

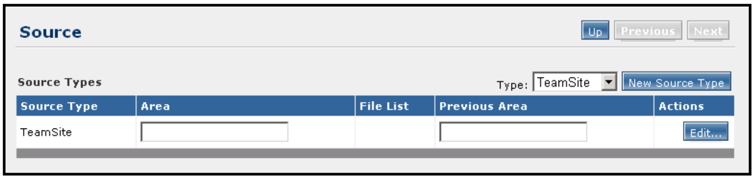
`/website/files` *or*

`C:\website\files`

The files in that location are compared with files residing on the target host, and the differences are deployed. See "Directory Comparison Deployments" on page 124 for more information.

TeamSite Comparison Deployments

For TeamSite comparison deployments, the Source window (Figure 83) displays attributes for the TeamSite area where the source files are located, and an alternate TeamSite area against which the first area will be compared and the differences deployed.



Source Type	Area	File List	Previous Area	Actions
TeamSite				Edit...

Figure 83: Source Window for TeamSite Comparison Deployments

By default, the Source window displays the attributes for a directory comparison deployment. To configure a TeamSite comparison deployment, you must change the Source window to display attributes for a TeamSite comparison deployment by selecting TeamSite from the **Type** list and clicking **New Source Type**. A source entry displaying the **Area** and **Previous Area** boxes is displayed. If you do not plan to use the existing **Filesystem** entry, remove it from the Source window by clicking its associated **Delete** button.

Enter the full path to the TeamSite area where your deployment's source files reside in the **Area** box. This TeamSite area can be a staging area, edition, or workarea. For example:

`//IWSERVER/default/main/dev/EDITION` *and*

Enter the full path to another TeamSite area on the source host against which the files residing in the **Area** box's location will be compared and the differences deployed. For example:

```
//IWSERVER/default/main/dev/EDITION/IW_PREV
```

See "TeamSite Comparison Deployments" on page 128 for more information.

File List Deployments

File list deployments are configured in the same Source window as is used for directory comparison deployments (Figure 82). File list deployments do not compare files. Instead, the files listed in a file list are deployed to the target. A file list deployment is configured similarly to a directory comparison deployment, except that the path to the file list on the source host is specified. The target host simply receives the files listed in the file list. See "File List Deployments" on page 135 for more information.

Enter the absolute path to the file system location where your deployment's source files reside in the **Area** box. For example:

`/website/files` *or*

`C:\website\files`

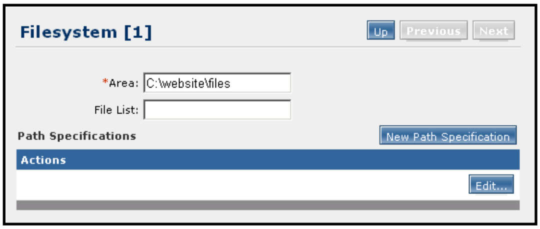
Enter the absolute path to the file list in the **Filelist** box of the **Filesystem** entry. For example:

```
C:\OpenDeploy\files\filelist.txt
```

Editing Source File Configurations

Each source file entry you create will have an entry in the tree. Select that entry from the tree, or click the **Edit** button associated with that entry in the Source window to display the corresponding source entry window.

Figure 84 displays a sample of a source file entry for a directory comparison.


Filesystem [1]

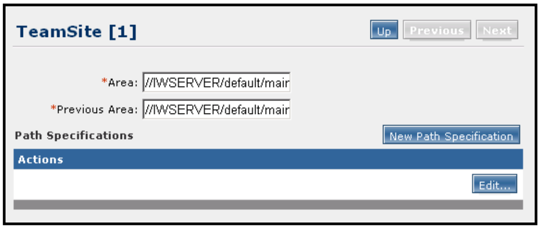
*Area: C:\website\files

File List:

Path Specifications

Figure 84: Filesystem Window for Editing a Directory Comparison Source File Entry

Figure 85 displays a sample of a source file entry for a TeamSite comparison.



TeamSite [1]

*Area: //IWSERVER/default/mair

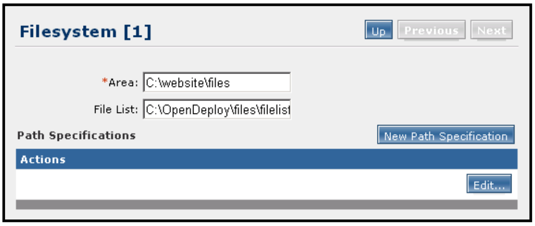
*Previous Area: //IWSERVER/default/mair

Path Specifications

Actions

Figure 85: TeamSite Window for Editing a TeamSite Comparison Source File Entry

Figure 86 displays a sample of a source file entries for a file list deployments.



Filesystem [1]

*Area: C:\website\files

File List: C:\OpenDeploy\files\filelist

Path Specifications

Actions

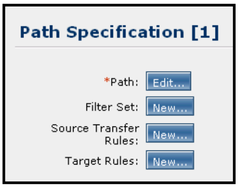
Figure 86: Filesystem Window for Editing a File List Source File Entry

Configuring Multiple Source Entries

You can configure more than one source entry within a definition. However, by default, all source entries deploy to the same target location. You run the risk of overwriting your files if you have multiple sources deploying simultaneously to the same target location. Multiple sources can also cause problems if one or more have the `doDeletes` feature enabled. In most cases, you only want to configure a single source entry. The one exception is when you use the Target Rules feature to define a different target location. See “Defining Source-Based Overrides” on page 139 for more information.

Defining a Subdirectory Within the Source File Location

In some cases, you might want to further narrow the source file location for you deployment within the specified source file system location or TeamSite area. You can specify a subdirectory relative to the source file area by entering that directory path in the Path Specification window (Figure 87).



Path Specification [1]

*Path: Edit...

Filter Set: New...

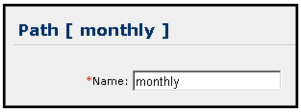
Source Transfer Rules: New...

Target Rules: New...

Figure 87: Path Specification Window

Select the **Path Specification** entry associated with your **Source** entry in the tree to display the corresponding Path Specification window. You can also click the **Edit** button in the corresponding source file entry (Figure 84).

Click **Path** to display the Path window (Figure 88).



Path [monthly]

*Name: monthly

Figure 88: Path Window

Enter the relative path within the specified file system location or TeamSite area. For example, if you wanted to deploy files from the directory monthly within the specified area, you would enter the following value in the **Name** box:

```
monthly
```

If you do not want to specify a subdirectory within your specified area, simply enter a period (".") in the **Name** box.

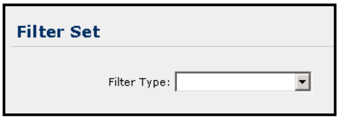
You can specify additional subdirectories within your source file area by clicking the **New Path Specification** button in your file system or TeamSite window. Each path specification entry you add will have a corresponding path element that you can access by clicking the corresponding **Edit** button.

Each path entry you add is displayed in the tree as a separate **Path** entry. Select the **Path** entry or click the **Edit** button associated with your path entry in the Path Specification window to display the Path window. You can apply filters that will exclude specified files and directories from deployments at the target (see next section).

Applying Source-Side Filters

You can include filters that will exclude files and directories at the source host, based on patterns in their paths names or their file names. See "Filters" on page 171 for more information on this feature.

Source-side filters are configured in the Filter Set window (Figure 89).



Filter Set

Filter Type:

Figure 89: Filter Set Window

Click the **New** button associated with **Filter Set** in the Path Specification window to display the Filter Set window. If filters are already configured, you can click the **Edit** button in the Path Specification window or select **Filter Set** from the tree to display the Filter Set window.

You can configure the following types and combinations of filters in the Filter Set window:

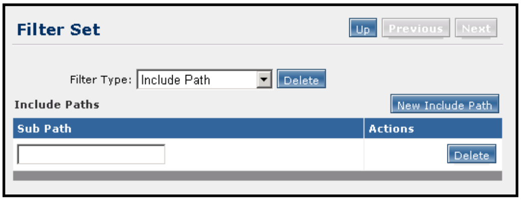
- File system-based inclusion filters — filters that only include those items whose paths match one or more of the path values you specify. File system-based inclusion filters cannot be used in combination with any other filter type.
- Pattern-based inclusion filters — filters that only include those items whose names match one or more of the regular expression values you specify. Pattern-based inclusion filters cannot be used in combination with any other filter type.
- File system-based exclusion filters — filters that exclude those items whose paths match one or more of the path values you specify. File system-based exclusion filters can be used in any combination with pattern-based exclusion filters. They can also be used in combination with either file system- or pattern-based exception filters, but not both.

- Pattern-based exclusion filters — filters that exclude those items whose names match one or more of the regular expression values you specify. Pattern-based exclusion filters can be used in any combination with pattern-based exclusion filters. They can also be used in combination with either file system- or pattern-based exception filters, but not both.
- File system-based exception filters — filters that protect those items whose paths match one or more of the path values you specify from any exclusion filters that would otherwise eliminate the items from the deployment. They can be used in combination with both file system- and pattern-based exclusion filters, but not with pattern-based exception filters.
- Pattern-based exception filters — filters that protect those items whose names match one or more of the regular expression values you specify from any exclusion filters that would otherwise eliminate the items from the deployment. They can be used in combination with both file system- and pattern-based exclusion filters, but not with pattern-based exception filters.

You can add file system- or pattern-based inclusion filters by selecting **Include Path** or **Include Pattern**, respectively, from the **Filter Type** list in the Filter Set window. You can add exclusion and exception filters by selecting **Exclude and Except**. The attributes associated with your selection are displayed in the Filter Set window.

File System-Based Inclusion Filters

Select **Include Path** from the **Filter Type** list to add a file system-based inclusion filter. Click **Include New Path** to display an **Include Path** filter entry (Figure 90).



Filter Set

Filter Type: Include Path

Include Paths

Sub Path	Actions
	Delete

Figure 90: File System-Based Inclusion Filter

Enter the path that all items must match to be included in the deployment in the **Sub Path** box. For example, to add a filter that would only include the contents of the directory `reports/monthly` within the specified area, enter the following value:

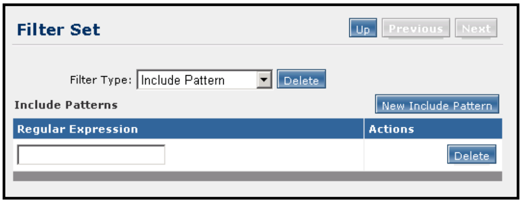
```
reports/monthly
```

The path you enter is relative to the local directory or TeamSite area on the source host containing the files to be moved. You can add additional file system-based inclusion filters by clicking **New Include Path** for each entry. You can delete any entry by clicking the associated **Delete** button.

You cannot use file system-based inclusion filters in combination with any other type of inclusion, exclusion, or exception filter.

Pattern-based Inclusion Filters

Select **Include Pattern** from the **Filter Type** list to add a pattern-based inclusion filter. Click **Include New Pattern** to display an **Include Pattern** filter entry (Figure 91).



Filter Type: Include Pattern

Regular Expression	Actions
	Delete

Figure 91: Pattern-Based Inclusion Filter

Enter the regular expression that all items' names must match to be included in the deployment in the **Regular Expression** box. For example, to add a filter that only includes files that end with the extension `.txt` within the specified area, enter the following value:

```
.*\.txt$
```

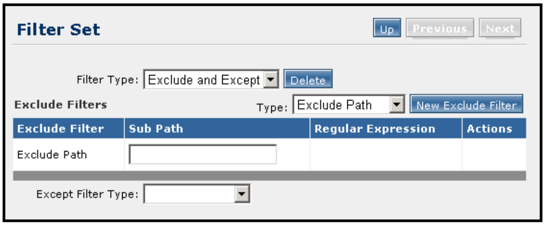
See "Supported Regular Expressions" on page 182 for more information on using regular expressions in deployment configurations.

You can add additional file system-based inclusion filters by clicking **New Include Pattern** for each entry. You can delete any entry by clicking the associated **Delete** button.

You cannot use pattern-based inclusion filters in combination with any other type of inclusion, exclusion, or exception filter.

File System-Based Exclusion Filters

Select **Exclude and Except** from the **Filter Type** list add any exclusion and exception filters. A file system-based exclusion filter entry is displayed by default (Figure 92).



Filter Set

Filter Type: Exclude and Except

Exclude Filters

Type: Exclude Path

Exclude Filter	Sub Path	Regular Expression	Actions
Exclude Path			

Except Filter Type:

Figure 92: File System-Based Exclusion Filter

Enter the path that all items must match to be excluded from the deployment in the **Sub Path** box. For example, to add a filter that ignores the directory `WebFiles/working` within the specified area, enter the following value:

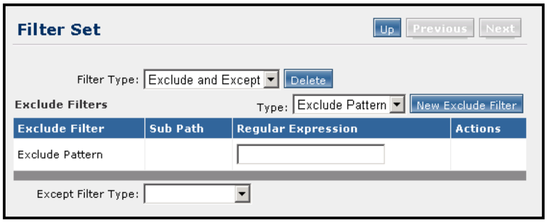
```
WebFiles/working
```

The path you enter in the **Sub Path** box is relative to the local directory or TeamSite area on the source host containing the files to be moved. You can add additional file system-based exclusion filters by clicking **New Exclude Path** for each entry. You can delete any entry by clicking the associated **Delete** button.

You can use file system-based exclusion filters with other types of exclusion filters. You can also use them with either file system- or pattern-based exception filters, but not both. You cannot use pattern-based exclusion filters with any type of inclusion filter.

Pattern-Based Exclusion Filters

Select **Exclude and Except** from the **Filter Type** list add any exclusion and exception filters. Select **Exclude Pattern** from the **Type** list and click **New Exclude Filter** to add a pattern-based exclusion filter entry (Figure 93).



Filter Set

Filter Type: Exclude and Except

Exclude Filters

Type: Exclude Pattern

Exclude Filter	Sub Path	Regular Expression	Actions
Exclude Pattern			

Except Filter Type:

Figure 93: Pattern-Based Exclusion Filter

Enter the regular expression that all items' names must match to be excluded from the deployment in the **Regular Expression** box. For example, to add a filter that ignores any file that ends with the extension `.html` within the specified area, enter the following value:

```
.*\.html$
```

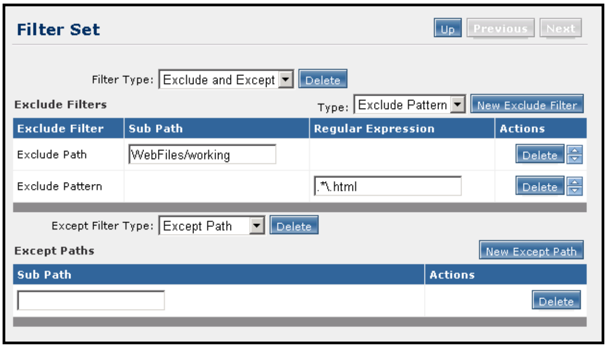
See "Supported Regular Expressions" on page 182 for more information on using regular expressions in deployment configurations.

You can add additional pattern-based exclusion filters by clicking **New Exclude Path** for each entry. You can delete any entry by clicking the associated **Delete** button.

You can use pattern-based exclusion filters with other types of exclusion filters. You can also use them with either file system- or pattern-based exception filters, but not both.You cannot use pattern-based exclusion filters with any type of inclusion filter.

File System-Based Exception Filters

Select **Except Path** from the **Except Filter Type** list displayed your exclusion filters to add a file system-based exception filter (Figure 94).



Filter Type: Exclude and Except

Exclude Filters

Type: Exclude Pattern

Exclude Filter	Sub Path	Regular Expression	Actions
Exclude Path	WebFiles/working		Delete
Exclude Pattern		.*\.html	Delete

Except Filter Type: Except Path

Except Paths

Sub Path	Actions
	Delete

Figure 94: File System-Based Exception Filter

Enter the path that all items must match to be protected from any exclusion filters in the **Sub Path** box. For example, to add a filter that protects the directory `WebFiles/reports/external` within the specified area, then you would enter the following value:

```
WebFiles/reports/external
```

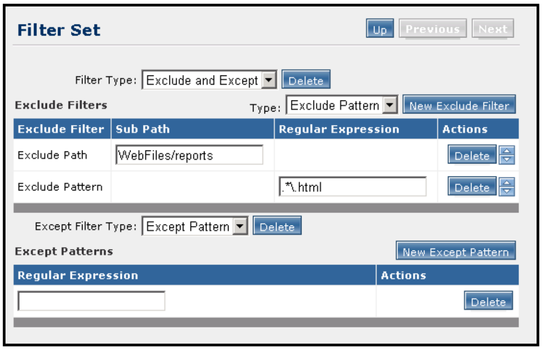
The path you enter in the **Sub Path** box is relative to the local directory or TeamSite area on the source host containing the files to be moved.

You can add additional file system-based exception filters by clicking **New Except Path** for each entry. You can delete any entry by clicking the associated **Delete** button.

You can use file system-based exception filters with any combination of exclusion filters. You cannot use them with other types of exception filters, or any type of inclusion filters.

Pattern-Based Exception Filters

Select **Except Pattern** from the **Except Filter Type** list displayed your exclusion filters to add a pattern-based exception filter (Figure 95).



Filter Set

Filter Type: Exclude and Except

Exclude Filters

Type: Exclude Pattern

Exclude Filter	Sub Path	Regular Expression	Actions
Exclude Path	WebFiles/reports		Delete
Exclude Pattern		.*\.html	Delete

Except Filter Type: Except Pattern

Except Patterns

Regular Expression	Actions
	Delete

Figure 95: Pattern-Based Exception Filter

Enter the regular expression that all items’ names must match to be protected from any exclusion filters in the **Regular Expression** box. For example, to add a filter that protects any files with the extension `.pdf`, enter the following value:

```
.*\.pdf$
```

See “Supported Regular Expressions” on page 182 for more information on using regular expressions in deployment configurations.

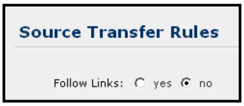
You can add additional pattern-based exception filters by clicking **New Except Pattern** for each entry. You can delete any entry by clicking the associated **Delete** button.

You can use pattern-based exception filters with any combination of exclusion filters. You cannot use them with other types of exception filters, or any type of inclusion filters.

Following Source-Side Symbolic Links in Deployments

You can configure a deployment running on a UNIX server to deploy files referenced in symbolic links, a procedure known as *following links*. This feature is not available with OpenDeploy running on Windows. See “Deploying Symbolic Links” on page 197 for more information on this feature.

You can configure your UNIX OpenDeploy host to follow links on the source side by enabling the feature in the Source Transfer Rules window (Figure 96).



Source Transfer Rules

Follow Links: ○ yes ● no

Figure 96: Source Transfer Rules Window

You can display the Source Transfer Rules window by clicking the associated **New** or **Edit** button in the Path Specification window (Figure 87). You can also select **Source Transfer Rules** from the tree if an entry already exists.

Select the **yes** option in the Source Transfer Rules window to enable the Follow Links feature.

You can also enable the follow links feature at the target side by enabling the **Follow Links** attribute in the Transfer Rules window. See “Applying Transfer Rules” on page 354 for more information.

Designating Alternate Targets from the Source

You can associate a target area with a particular source in a file system comparison deployment with multiple sources. This feature provides the ability to create multiple source/target pairings for a single deployment. See “Defining Source-Based Overrides” on page 139 for more information on this feature.

You can designate an alternate target location for a set of deployed files through the Target Rules window (Figure 97).

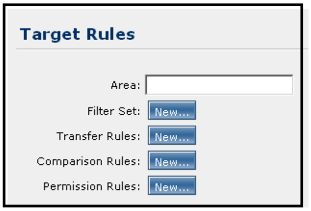


Figure 97: Target Rules Window

You can display the Target Rules window by clicking the associated **New** or **Edit** button in the Path Specification window (Figure 87). You can also select **Target Rules** from the tree if an entry already exists.

The Target Rules window contains the **Area** box, where you can enter the path for an alternate target file location. There are also buttons for other features that apply only to those files that are being deployed. Each of these features is described separately.

Enter the full path to the alternate target location for the source files that use this feature. For example, if you wanted to deploy to the following location:

```
D:\metadata\files
```

enter that path in the **Area** box.

Other features accessible in the Target Rules window that you can apply to alternate targets are listed below:

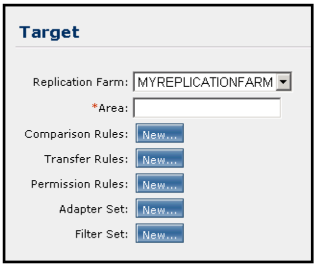
- **Filter Set** — see “Applying Target-Side Filters” on page 361.
- **Transfer Rules** — see “Applying Transfer Rules” on page 354.
- **Comparison Rules** — see “Applying Comparison Rules” on page 353.
- **Permission Rules** — see “Applying Permission Rules” on page 356.

Defining the Target File Location

The target file location is the file system location on the target host where deployed files reside following the deployment. In directory comparison deployments, any files residing in the target directory location are compared with equivalent files in the source file location, and the differences (based on the criteria for deployment) are deployed to the target. In TeamSite comparison and file list deployments, the target file location is simply a repository for the deployed files. Any files residing in the target file location are excluded from the comparison of TeamSite areas on the source host. Files residing in the target file area prior to running the deployment can be overwritten by like-named deployed files.

By default, one target directory location is associated with each definition element in a deployment configuration. If you have multiple sources within a definition, they will (by default) deploy files to the same target directory location. You can use the target rules option to create alternate target directory locations on a source-by-source basis. See "Designating Alternate Targets from the Source" on page 350 for more information on that feature.

The target file location is defined in the Target window (Figure 98).



Target

Replication Farm: MYREPLICATIONFARM

*Area:

Comparison Rules: New...

Transfer Rules: New...

Permission Rules: New...

Adapter Set: New...

Filter Set: New...

Figure 98: Target Window

You can display the Target window by clicking the associated **Edit** button in the Definition window (Figure 81), or by selecting the **Target** entry in the tree.

Each target must be associated with a particular replication farm element. The replication farm contains one or more individual target host nodes that will receive the deployed files. The information you enter into the Target window determines the target file location on the target host nodes making up the members of the replication farm. If you have more than one replication farm element defined in your configuration, select the appropriate choice from the **Replication Farm** list.

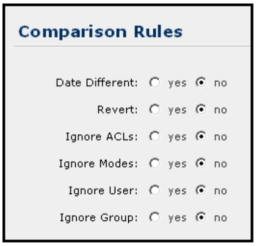
Enter the full path to the target file location in the **Area** box. This value must be a file system location. You cannot enter a TeamSite area. If you want to deploy files into a TeamSite workarea, enter that location as a file system path, for example:

```
Y:\default\main\dev\WORKAREA\jdoe
```

Applying Comparison Rules

Comparison rules define the rules that OpenDeploy uses when it compares files contained in a file system. The type and platform of the file system determines the criteria available. These rules are used to determine eligibility of files for deployment. See "File Comparison Rules" on page 184 for more information on this feature.

Comparison rules are defined in the Comparison Rules window (Figure 99).



Comparison Rules

Date Different: ○ yes ● no

Revert: ○ yes ● no

Ignore ACLs: ○ yes ● no

Ignore Modes: ○ yes ● no

Ignore User: ○ yes ● no

Ignore Group: ○ yes ● no

Figure 99: Comparison Rules Window

You can display the Comparison Rules window by clicking the **New** or **Edit** button associated with Comparison Rules in the Target window (Figure 98), or by selecting its entry from the tree.

The Comparison Rules window contains the following options:

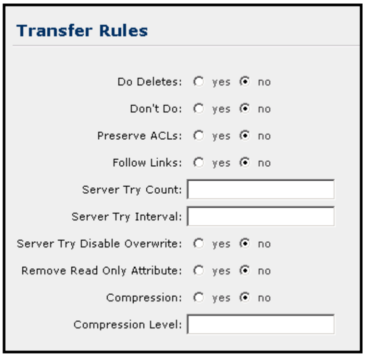
- **Date Different** option — indicate whether or not a file should be deployed if there is any difference in file date (older or newer) between the source and target versions. This differs from the OpenDeploy default date-based comparison setting, where a file should be deployed only if the source file is newer than the target file.
- **Revert** option — indicate whether or not a file should be deployed if the source version is older than the target version.

- **Ignore ACLs** option (Windows only) — indicate whether (**yes**) or not (**no**) to ignore differences in the Windows access control lists (ACLs) during the file comparison.
- **Ignore Modes** option (UNIX only) — indicate whether (**yes**) or not (**no**) to ignore differences in the UNIX-based permission mode bits definitions during the file comparison.
- **Ignore User** option (UNIX only) — indicate whether (**yes**) or not (**no**) to ignore differences in the UNIX-based user ownership during the file comparison.
- **Ignore Groups** (UNIX only) option — indicate whether (**yes**) or not (**no**) to ignore differences in the UNIX-based group ownership during the file comparison.

Applying Transfer Rules

Transfer rules define the rules for moving files from the source host to the target host during the deployment. See "File Transfer Rules" on page 186 for more information on this feature.

Transfer rules are defined in the Transfer Rules window (Figure 100).



Transfer Rules

Do Deletes:	○ yes ● no
Don't Do:	○ yes ● no
Preserve ACLs:	○ yes ● no
Follow Links:	○ yes ● no
Server Try Count:	
Server Try Interval:	
Server Try Disable Overwrite:	○ yes ● no
Remove Read Only Attribute:	○ yes ● no
Compression:	○ yes ● no
Compression Level:	

Figure 100: Transfer Rules Window

You can display the Transfer Rules window by clicking the **New** or **Edit** button associated with Transfer Rules in the Target window (Figure 98), or by selecting its entry from the tree.

The Transfer Rules window contains the following options:

- **Do Deletes** option — select whether (**yes**) or not (**no**) files and directories not present in the source host area will be deleted on the target host.
- **Don't Do** option — select whether (**yes**) or not (**no**) to proceed with the deployment following the comparison. Deployment will not occur if this attribute is enabled. This is a good tool to use to check and compare files without actually performing a deployment.
- **Preserve ACLs** option (Windows only) — select whether (**yes**) or not (**no**) to preserve the Windows access control lists (ACLs) when the files are moved.
- **Follow Links** option — select whether (**yes**) or not (**no**) symbolic links on the target hosts will be followed when the files are moved.
- **Server Try Count** box (Windows only) — enter the number of times OpenDeploy will attempt to deploy the file to the target host. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.
- **Server Try Interval** box (Windows only) — enter the amount of time in seconds OpenDeploy waits between deployment attempts. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.
- **Server Try Disable Overwrite** option (Windows only) — select whether (**yes**) or not (**no**) to disable the ability of OpenDeploy to deploy files to a server even if the `svrTryCount` and `svrTryInterval` elements are specified. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.
- **Remove Read Only Attribute** — select whether (**yes**) or not (**no**) a deployed file can overwrite its read-only target equivalent. If enabled, OpenDeploy removes the read-only attribute from the target file, allowing the deployment to occur.
- **Compression** — select whether (**yes**) or not (**no**) to indicate whether content is compressed prior to be deployed.
- **Compression Level** — enter the level of compression (0-9) OpenDeploy uses when compression is enabled. A value of 1 is the lowest level of compression, and 9 is the highest level. A value of 0 provides no compression at all.

Applying Permission Rules

Permission rules define the rules applicable to the permissions of deployed files and directories. See “File Permission Rules” on page 190 for more information on this feature.

Permission rules are defined in the Permission Rules window (Figure 101).

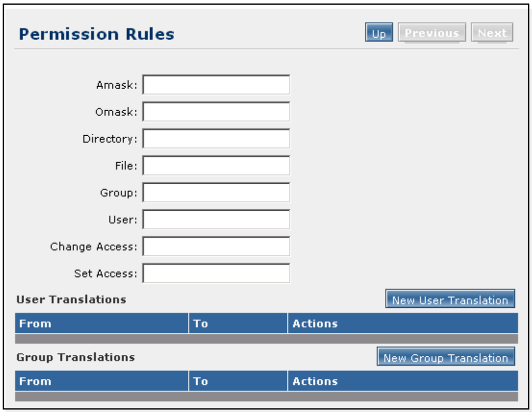


Figure 101: Permission Rules Window

You can display the Permission Rules window by clicking the **New** or **Edit** button associated with Permission Rules in the Target window (Figure 98), or by selecting its entry from the tree.

Access options specific to UNIX are ignored when deploying to a Windows target host, and access options specific to Windows are ignored when deploying to a UNIX target host.

The Permission Rules window contains the following options:

- **Amask** box (UNIX only) — enter the bit mask (in octal) to be ANDed with the permission bits of all files and directories. The `amask` octal value combines with the existing permission bit value of the affected file. If a file has the existing permission value of `664` (-rw-rw-r--) and the `amask` attribute as the following value:

  ```
  amask="770"
  ```

  then the resulting permission for that file (664 AND 770) following the deployment would be `660` (-rw-rw----).

- **Omask** box (UNIX only) — enter the bit mask (in octal) to be ORed with the permission bits of all files and directories. The `omask` octal value combines with the existing permission bit value of the affected file. If a file has the existing permission value of `666` (-rw-rw-rw-) and the `omask` attribute as the following value:

  ```
  omask="022"
  ```

  then the resulting permission for that file (666 OR 022) following the deployment would be `644` (-rw-r--r--).

- **Directory** box (UNIX only) — enter the permissions (in octal) given to all deployed directories. For example, if you wanted deployed directories to have the permission "drwxrwx---", then the resulting value would be:

  ```
  directory="770"
  ```

- **File** box (UNIX only) — enter the permissions (in octal) given to all deployed files. For example, if you wanted deployed files to have the permission "-rw-rw-r-x" , then the resulting value would be:

  ```
  file="665"
  ```

- **Group** box (UNIX only) — enter the group assigned to all deployed files and directories. This attribute value must be a valid group name or group ID. For example:

  `group="tech_pubs"` *or*

  `group="200"`

  You must also specify the `user` attribute if you use employ the `group` attribute.

- **User** box (UNIX only) — enter the user who will own all deployed files and directories. This attribute value must be a valid user name or user ID. For example:

  `jdoe` *or*

  `105`

  You must also specify the group attribute if you use employ the user attribute.

- **Change Access** box (Windows only) — enter a value that modifies the access control lists (ACLs) so that specified users have the designated rights. The new access control entry (ACE) for each specified user allows only the specified rights, discarding any existing ACE. In the following example:

  ```
  changeAccess="{ jdoe:W, tech_pubs:NONE }"
  ```

  any existing ACEs for `jdoe` and `tech_pubs` are removed, `jdoe` is granted write access, and the group `tech_pubs` has no access at all. Any other access rights that may have existed for other users are left unchanged. See "Using OpenDeploy with ACLs" on page 194 for more information.

- **Set Access** box (Windows only) — enter a value that replaces the ACLs for the deployed files and directories. In the following example:

  ```
  setAccess="{ jdoe:ALL, tech_pubs:RX }"
  ```

  the existing ACL is removed and the user `jdoe` is granted full access. The group `tech_pubs` has read access to the specified files. Any other access rights that may have existed for the file are removed. See "Using OpenDeploy with ACLs" on page 194 for more information.

User Translation

The User Translation feature (Figure 102) allows you to change user IDs (UNIX only) for deployed files. You can add and configure an unlimited number of user translations in your deployment. See "User and Group Ownership Transferal" on page 193 for more information on this feature.

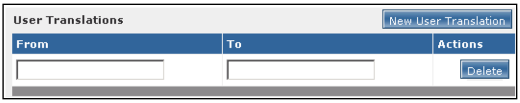


Figure 102: User Translation Feature

The User Translation feature contains the following attributes:

- **New User Translation** button — click to add a user translation entry.

- **From** box — enter the existing source user or user ID (the identification number assigned to each user account within the UNIX server). For example:

  `jdoe` *or*

  `105`

- **To** box — enter the new target user or user ID. For example:

  `rroe` *or*

  `110`

- **Delete** button — click to delete a user translation entry.

Group Translation

The Group Translation feature (Figure 103) is where you can change group IDs (UNIX only) for deployed files. You can add and configure an unlimited number of group translations in your deployment. See "User and Group Ownership Transferal" on page 193 for more information on this feature.

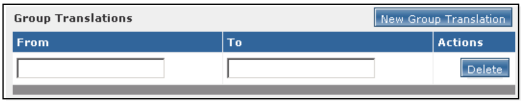


Figure 103: Group Translation Feature

The Group Translation feature contains the following attributes:

- **New Group Translation** button — click to add a group translation entry.
- **From** box — enter the existing source group or ID (the identification number assigned to each group account within the UNIX server). For example:

  `tech_pubs` *or*

  `100`

- **To** box — enter the new target group or ID. For example:

  `marketing` *or*

  `200`

- **Delete** button — click to delete a group translation entry.

Configuring for Use with Adapters

You can configure the deployment for the enabling of an adapter (Java program) written to run in the Delivery Adapter Framework. A base server host can be configured to load and run the designated Java class (adapter) upon completion of an deployment it receives. After the content is received, the adapter can push the content somewhere else, such as to an edge server over HTTP. This functionality is available only for targets running the base server software, not the receiver software. See Appendix A, "Java Adapters" on page 365 for more information on this feature.

You can configure the enabling of an adapter in the Adapter Set window (Figure 104).

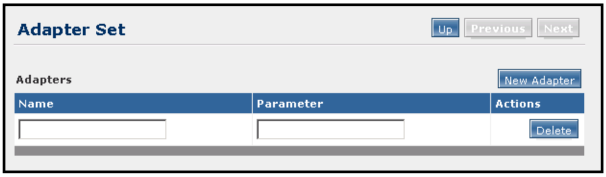


Figure 104: Adapter Set Window

You can display the Adapter Set window by clicking the **New** button associated with **Adapter Set** in the Target window.

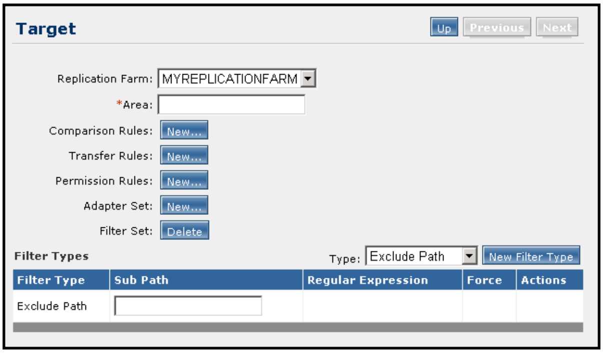
The Adapter Set window contains the following attributes:

- **New Adapter** button — click to display the attributes for the adapter.
- **Name** box — enter the name of the Java class that implements the adapter.
- **Parameter** box — specify the parameter for the adapter.
- **Delete** button — click to delete the entry.

Applying Target-Side Filters

You can include filters that will exclude files and directories from being received by the target host, based on patterns in their paths names or their file names. See “Filters” on page 171 for more information on this feature.

Click **Filter Set** in the Target Rules window to display the filtering configuration information (Figure 105):



Target

Replication Farm: MYREPLICATIONFARM

*Area:

Comparison Rules:

Transfer Rules:

Permission Rules:

Adapter Set:

Filter Set:

Filter Types

Type: Exclude Path

Filter Type	Sub Path	Regular Expression	Force	Actions
Exclude Path				

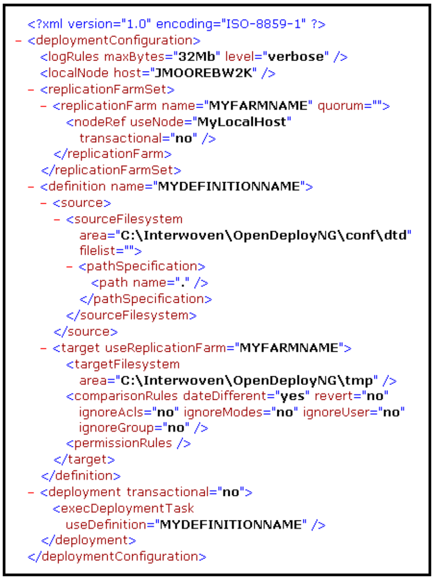
Figure 105: Target Window With Filtering Attributes Displayed

Applying path- and pattern-based inclusion and exclusion filters to a target is similar to applying them to the source host. See “Applying Source-Side Filters” on page 343 for more information on filter usage.

Saving the Deployment

When you have completed configuring your deployment, you must save it by clicking the **Save** button at the top of the Deployment Configuration Composer window. If any required items of information are missing, such as required attribute values, the Deployment Configuration Composer will display the appropriate error messages in the Error pane. See “Details Pane” on page 316 for an example.

Upon successfully saving the deployment, the configuration is displayed (Figure 106).



```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <deploymentConfiguration>
    <logRules maxBytes="32Mb" level="verbose" />
    <localNode host="JMOOREBW2K" />
  - <replicationFarmSet>
    - <replicationFarm name="MYFARMNAME" quorum="">
        <nodeRef useNode="MyLocalHost"
          transactional="no" />
      </replicationFarm>
    </replicationFarmSet>
  - <definition name="MYDEFINITIONNAME">
    - <source>
      - <sourceFilesystem
          area="C:\Interwoven\OpenDeployNG\conf\dtd"
          filelist="">
        - <pathSpecification>
            <path name="." />
          </pathSpecification>
        </sourceFilesystem>
      </source>
    - <target useReplicationFarm="MYFARMNAME">
        <targetFilesystem
          area="C:\Interwoven\OpenDeployNG\tmp" />
        <comparisonRules dateDifferent="yes" revert="no"
          ignoreAcls="no" ignoreModes="no" ignoreUser="no"
          ignoreGroup="no" />
        <permissionRules />
      </target>
    </definition>
  - <deployment transactional="no">
      <execDeploymentTask
        useDefinition="MYDEFINITIONNAME" />
    </deployment>
  </deploymentConfiguration>
```

Figure 106: Example of Saved Deployment

Editing Deployment Configurations

You can use the Deployment Configuration Composer to edit existing configurations, even those that were not created using this tool. You can edit the configuration of any deployment that resides in the *od-admin*/conf directory, and that conforms to XML-based structure used in OpenDeploy 5.x deployment configurations.

To edit an existing deployment configuration, follow these steps:

1. Select **Configurations > View Configurations** to display the Deployment Configuration window.
2. Select the OpenDeploy host on which the deployment configuration will reside from the **Selected Host** list.
3. Select the deployment you want to edit from the **Deployment** list.
4. Click **Edit** to open a new browser window containing the containing the Deployment Configuration Composer. The elements and attributes for that deployment are displayed in the Details pane.
5. Modify your deployment using the methods described in this chapter.
6. Click **Save** to save your changes.

Editing Configuration From Previous OpenDeploy Releases

The Deployment Configuration Composer only generates deployment configurations compatible with this release of OpenDeploy. You can open deployment configurations from previous OpenDeploy 5.x releases, but they will be saved as files compatible for this release only.

If you want to modify a deployment configuration for use with an older version of OpenDeploy, you should open the file with a text or XML editor and make your changes manually.

Appendix A

Java Adapters

This appendix lists and describes the Java adapters that have been tested and approved for use with OpenDeploy as part of the Delivery Adapter Framework.

Sample Adapter

OpenDeploy provides a sample Java adapter that generates a log file that contains all the manifest information about the directories and files that are deployed or deleted when the deployment is run. This sample adapter is intended to illustrate how you can implement your own adapters within the Delivery Adapter Framework. The required Java code for use with this sample adapter resides in your host at the following location:

```
od-home/solutions/adapter/example/AdapterExample.java
```

To incorporate this functionality to your deployment, add the following code to the deployment configuration within the `target` element:

```
<odAdapterSet>
   <odAdapter name="AdapterExample" parameter="" />
</odAdapterSet>
```

When the deployment is run, the information on the deployed directories and files is written to the following file:

```
od-home/log/hostname_adapter.log
```

where *`hostname`* is the name of the source host. For example, `mars_adapter.log`.

The contents of the manifest log file are based on the structure specified in the internal manifest DTD file. You can access this DTD at:

```
od-home/conf/dtd/odmanifest.dtd
```

Refer to Chapter 8, "OpenDeploy Manifest DTD" on page 197 in the *OpenDeploy Installation and Reference Guide* for more information on the internal manifest DTD.

Network Appliance NetCache Adapter

OpenDeploy includes a Network Appliance NetCache adapter that can be integrated into the Delivery Adapter Framework. When configured to use this adapter, an OpenDeploy base server host can receive deployed content and push this content to the NetCache devices over HTTP. No OpenDeploy software is needed on the NetCache devices.

Figure 107 shows how content is moved from the development source host to the edge servers.

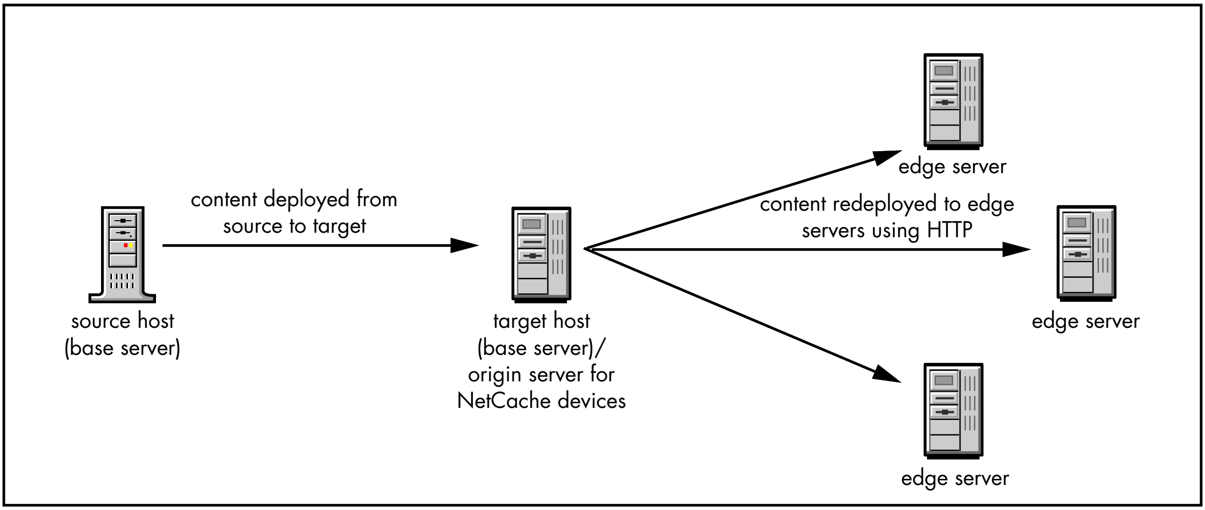

content deployed from source to target
source host (base server)
target host (base server)/ origin server for NetCache devices
content redeployed to edge servers using HTTP
edge server
edge server
edge server

Figure 107: Deploying Content to Edge Servers

Using this method, you can implement a single, consistent, and global solution for distributing content from development to production, and on to the very edge of the network.

OpenDeploy provides the files necessary to configure and run this type of deployment. These files reside in the following location:

```
od-home/solutions/adapter/netcache
```

NetCache Adapter Configuration File

OpenDeploy includes an XML-based configuration file for use with the NetCache adapter. For example:

```
<netCacheServerConfiguration>
   <serverSet>
      <server host="andromeda"
         port="3132" userid="jdoe" password="aBcDeF321"
         isPasswordEncrypted="yes" webServerName="NYweb1"
         logFile="/dir/logfile.txt">
         <cacheProperties minAge="60" maxAge="36000" lockTimeout="60" />
      </server>
   </serverSet>
</netCacheServerConfiguration >
```

This file resides in the following location:

```
od-home/solutions/adapter/netcache/NetCache.xml
```

You must open this file with a text or XML editor and customize the elements and attribute values to meet your enterprise's specifications:

The NetCache adapter configuration file contains a root-level `netCacheServerConfiguration` element that contains the elements and attributes that determine how deployed content will be pushed to the NetCache edge servers. Within the root element is the `serverSet` element, which is a container for the individual NetCache server configurations.

Each individual NetCache device is configured by the `server` element within the `serverSet` element. There must be a separate occurrence of the `server` element for each NetCache device to which deployed content is to be pushed. Within the `server` element, you must specify values for the following attributes:

- `host` — specify the NetCache host name or IP address.
- `userid` — specify the user ID for accessing the NetCache device.
- `password` — specify the password associated with the user ID.
- `isPasswordEncrypted` — indicate whether (`yes`) or not (`no`) the password is encrypted using B64 encoding. The default value is `no`.

- `webServerName` — specify the name of the Web server that acts as the origin server for the NetCache device.
- `logFile` — specify the path and file name of the log file for the adapter.
- `port` — specify the NetCache administration port.

With each `server` element, you have the option of specifying a `cacheProperties` element and some or all of its associated attributes. The `cacheProperties` element defines the properties associated with the NetCache server's cache. Here are the attributes that you can specify within the `cacheProperties` element:

- `minAge` — (optional) specify the number of seconds that the object in the cache is not visible to HTTP clients.
- `maxAge` — (optional) specify the number of seconds that the object in the cache is valid to HTTP clients.
- `lockTimeout` — (optional) specify the number of seconds that the object in the cache is locked in the cache.

Setup in Deployment Configuration

In your OpenDeploy deployment configuration, you must add the following `odAdapterSet` and `odAdapter` elements within your `target` element:

```
<target ...>
   <odAdapterSet>
      <odAdapter
         name="com.interwoven.od.adapter.netcache.IWODNetCache"
         parameter="od-home/conf/NetCache.xml"
         />
      ...
   </odAdapterSet>
</target>
```

Retrying When Push Fails But Overall Deployment Succeeds

In some cases, the pushing of content to edge serves can fail even if the rest of the deployment succeeded. If this occurs, you can retry the pushing of content to the edge servers without having to rerun the deployment by running the `iwodnetcache` command line tool.

There are various options associated with the `iwodnetcache` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodnetcache -h | -v
```

```
iwodnetcache -m manifest_file -p config_file -s source_area -d od_home
```

`-h`	Displays help information.
`-v`	Displays version information.
`-m` *`manifest_file`*	Specifies the absolute path to the manifest file. If a full path is not specified, the default directory is *`od-home`*`/log`.
`-p` *`config_file`*	Specifies the absolute path to the NetCache configuration file.
`-s` *`source_area`*	Specifies the content source area for the deployment to the NetCache edge servers.
`-d` *`od_home`*	Specifies the OpenDeploy home directory (*`od-home`*).

For example, if you entered the following command:

```
iwodnetcache -m rcv.deploy.DEF1.source.to.target.log.mf
-p /usr/local/OpenDeployNG/NetCache.xml -s /usr/webfiles
-d /usr/local/OpenDeployNG
```

The path to the manifest file would be:

```
od-home/log/rcv.deploy.DEF1.source.to.target.log.mf
```

The path to the NetCache configuration file would be:

```
/usr/local/OpenDeployNG/NetCache.xml
```

The content source area location for the content being deployed to the NetCache edge servers would be:

```
/usr/webfiles
```

The OpenDeploy home directory is:

```
/usr/local/OpenDeployNG
```

Internationalization

NetCache does not support internationalization. As a result, the OpenDeploy NetCache Adapter does not support internationalization.

Glossary

This glossary lists terms and their definitions found in this manual.

adapter	Java-based program that extends content distribution to specialized devices and protocols, such as edge or cache servers, within the Delivery Adapter Framework.
administration server	A server with the OpenDeploy administration software installed. The administration server is responsible for generating and managing the browser-based user interface in the OpenDeploy environment.
Administrator role	The highest level of OpenDeploy access. An individual with the Administrator role has the ability to configure OpenDeploy hosts, create and start deployment configurations, and set access restrictions for individuals in the User role.
area	A file system- or TeamSite-based location where files reside or will reside as the result of a deployment. Files residing in one area can also be compared with files in another area to determine whether they should be deployed.
asynchronous deployment	The practice of starting a deployment, but not waiting for the deployment to end before moving on to other tasks. When a deployment is run asynchronously, only the deployment's success or failure to start is returned. No indication of the deployment's success or failure to complete is presented. Asynchronous deployments are only available when using the `iwodstart` command-line tool.
attribute	A directive with a corresponding value that can be used to alter the default behavior of OpenDeploy, or that must be used to provide required information.
base server	The OpenDeploy software installed on a server that is licensed to send and receive deployments.
base server configuration file	An XML-based file that defines the rules and settings for a base server within the OpenDeploy environment. The base server configuration file must follow the XML rules as defined in the deploy server configuration DTD.

base server log	A log file containing entries related to the base server host.
bind port number	The number for the port that source and target hosts use to transport deployed files.
bootstrap administrator	The initial Administrator role user identity used to assign the Administrator role to other individuals.
certificate	A file which assures that both the source and target hosts in an SSL key encrypted deployment are certified to be taking part.
certificate authority	A set of programs used to generate public and private key pairs, and a database that contains state information. Certificate authority is used in conjunction with SSL encryption.
cipher	An encryption tool that the source and target hosts share to hide the identity of deployed files.
compare phase	The period of time during a deployment when files are being compared to determine whether they should be deployed. This is also the time during a deployment when it can be cancelled.
compression	The reduction of the size of files using compression algorithms. Compression results in a smaller deployment which speeds the transfer time to targets.
custom report	A method of constructing a report query through the browser-based user interface by entering or selecting search values.
definition	The matching of one or more source file locations (either file system locations or TeamSite areas) with a target file location (a file system location) for the purpose of deploying and receiving files. A deployment configuration can have one or more definitions between file locations on the sending host and the target file location.
Delivery Adapter Framework	A framework that enables the creation of application-specific adapters for extending the content delivery capabilities of OpenDeploy. This allows you to extend the reach of your content distribution network to specialized devices and protocols, such as edge or cache servers.
Deploy and Run	An OpenDeploy feature that allows external scripts or programs to be integrated into the deployment process.
deploy server configuration DTD	The XML-based DTD upon which the base server and receiver configuration files is derived.
deployment	The moving of files from a source host to one or more target hosts based on a particular deployment configuration.

deployment configuration	A combination of elements and attribute values which define the criteria for if and how files are to be deployed.
deployment configuration file	An XML-based file that defines the rules and settings for a source host to deploy files to one or more target hosts.
deployment criteria	The conditions that indicate whether a file should be deployed as part of a deployment configuration based on particular criteria.
development server	A server within the organizational firewall where content is developed and tested prior to being deployed to a production server.
directory comparison	A type of deployment configuration where an area on the source and target hosts are compared with each other, and the differences, based on a set of specified criteria, are deployed to the target host.
EasyDeploy	An alternate version of OpenDeploy without fan-out or multi-tiered deployment support, and no encryption.
edition	A TeamSite term for a snapshot of files for the purposes of archiving.
element	A logical unit of information within an XML-based document.
encryption	The ability to obscure the content of deployments moving between the source and target hosts to prevent unauthorized access.
exception filter	A filter used to protect files and directories based on their path or name from any exclusion filters that would otherwise omit them from the deployment.
exclusion filter	A filter used to exclude files and directories from a deployment based on their path or name.
fan-out deployment	A deployment configuration where the source host simultaneously deploys files to multiple target hosts.
file comparison rules	Optional criteria to use for determining whether to deploy files from the source host to the target host.
file list deployment	A deployment configuration where the source host deploys files to the target hosts based on a predetermined list of files.
file permission rules	Optional directives to apply to the deployed files or directories on the target host.
file transfer rules	Optional directives related to moving files from the source host to the target hosts.

filtering	Specification of directory paths or file name patterns to include with or exclude from a deployment.
group translation	The switching of UNIX-based group ownership on a deployed file or directory between the source host and the target host.
host	A server on which one or more OpenDeploy software components reside.
host reporting configuration file	A configuration file residing on each base server or receiver that determines how that host reports events within the reporting environment.
inclusion filter	A filter used to include files and directories in a deployment based on their path or name.
information stream	A listing of files and directories that are included in a deployment. This data can be streamed to a manifest- or log-based format using Deploy and Run.
instance	A particular running of a deployment configuration. An instance name can be appended to the deployment to differentiate it from other occurrences of the same deployment.
leg	The movement of deployed files from a sending host to a specific target.
log files	Files containing information related to the status of the OpenDeploy server or a particular deployment.
log format	A legacy format used to organize the information stream data.
logging level	One of the choices that determines the amount and type of logged information regarding deployments.
logical host name	A name mapped to a host's fully qualified DNS host name or IP address that is used to identify that host in configurations.
macro deployment log	The log file containing entries related to the activities involving a deployment configuration.
manifest format	An XML-based format used to organize the information stream data.
micro deployment log	The log file containing entries related to the activities involving each individual source/target pairing of a deployment.
multi-host installation	The installation of all the required OpenDeploy software components on multiple servers in some combination.
multi-tiered deployment	A deployment configuration where deployed files are received by an OpenDeploy base server host and then redeployed across *tiers*.

multi-tiered transactional deployment	A multi-tiered deployment in which the deployment transaction spans all servers.
node	A host that can send or receive files.
node configuration file	An XML-based configuration file that defines all the target hosts for a source host.
normal logging level	A logging level that logs standard status and error messages.
OpenDeploy Trial	An alternate version of OpenDeploy that provides full functionality for a limited period of time.
operations server	A server on which operations server software is installed. Operations server software determines roles and permissions of administrators and users within the OpenDeploy environment.
parameter substitution	A special key-value syntax that allows you to specify parameter values when starting or scheduling a deployment. Using this feature, you can specify different values for the same parameter each time you start a deployment.
path	An element that further defines a specified file system location or TeamSite area.
path-based filter	An inclusion or exclusion filter that compares the path of each file or directory with a specified path to determine whether or not the item can be deployed.
pattern-based filter	An inclusion or exclusion filter that compares the name of each file or directory with a specified regular expression to determine whether or not the item can be deployed.
physical host name	The fully qualified DNS host name or IP address of a host.
pre-commit phase	A period of time when a transactional deployment determines whether the deployment has been successful and can commit the deployment to all the targets. If the deployment cannot commit, the deployment is rolled back and the target hosts are restored to their previous states.
previous area	A second TeamSite area against which the primary area is compared in a TeamSite comparison deployment configuration. The previous area is typically the previous version of the files in the primary area, and also represents the files residing on the target hosts.

primary area	The TeamSite area in a TeamSite comparison that contains the most current files. The contents of the primary area are compared with those in the previous area to determine which files are deployable.
production server	Typically, a host with content that is accessible either on an intranet or the Internet.
quick report	A predefined query that can be accessed and run at any time without any additional report configuration required.
quorum	The specified minimum number of successful legs in a transactional deployment for the overall deployment to be considered successful.
receiver	A host on which OpenDeploy receiver software is installed. A receiver can only receive deployed files from a source host.
receiver log	A log file containing entries related to the receiver host.
receiver macro deployment log	A log generated by the receiving host that contains information for the received deployment.
receiver micro deployment log	A log generated by the receiving host that contains information regarding a specific source/target pairing in a deployment.
recurring deployment	A scheduled deployment where the deployment repeats on a regular basis, such as daily or weekly.
reporting	The collection and displaying of deployment information that is managed by the reporting server in a central database.
reporting management configuration file	A configuration file used by the reporting server to subscribe to deployment events from base server and receiver hosts.
reporting server	Software that runs the OpenDeploy reporting feature. The reporting server is co-located with the administration server software.
reverse deployment	A deployment configuration where files residing on a target host are deployed back to the source host.
reverse source	The sender of a reverse deployment.
reverse target	The recipient of a reverse deployment.
revert	File comparison attribute where a file is deployed if the source version is older than the target version.

roles	The collective term for the Administrator and User roles. The role of an individual user determines what level of features and functionality that person has access to within the OpenDeploy user interface.
rollover threshold	The size a log file can grow before it is closed to new entries and archived. A new log file is then generated.
schedule	A predetermined time and date when a particular deployment is started. This can occur on a one-time only or recurring basis.
scheduled deployment	A deployment that is performed at a predetermined time and date. A scheduled deployment can occur on a one-time or recurring basis.
scheduler database	A database that is installed with the base server software and stores the scheduling information for the source host.
service configuration file	A configuration file located on an OpenDeploy host with the base server or receiver software installed that specifies the name of the base server, receiver, and nodes configuration file being accessed by OpenDeploy. The service configuration file is named `deploy.cfg`.
simulated deployment	A deployment where no files are moved, but entries are made in the deployment logs for every file or directory that would have been deployed. This feature can be used to determine differences in files on the source and target hosts without actually deploying files.
single-host installation	The installation of all the required OpenDeploy software components on a single server.
source host	A host with base server software installed and licensed that can deploy and receive files.
source macro deployment log	A log generated by the sending host that contains information for the sent deployment.
source micro deployment log	A log generated by the sending host that contains information on a specific deployment source/target pairing. If the deployment moves files to multiple target hosts, each source/target pairing will have its own micro deployment log.
SQL query report	A method of constructing a free-form report query.
SSL key encryption	A high level (up to 168-bit) of file encryption you can assign to deployed files, which requires setting up a Secure Sockets Layer (SSL) certificate authority and generating the certificate.

staging area	A TeamSite area that receives and stores files submitted to it from the *workareas* it supports. There is a single staging area for each TeamSite branch.
submit	A TeamSite task action where files are moved from a workarea to the staging area.
successful deployment	A deployment that either successfully moved the deployed files to all of its intended targets, or at least to the number of targets specified by the quorum value.
symmetric key encryption	A lower level (40-bit) of encryption you can assign to deployed files.
synchronized deployment	The distribution of file and database assets together using OpenDeploy and DataDeploy in a single deployment transaction.
target host	A host with either receiver or base server software installed, which can receive deployed files from a source host.
TeamSite	Interwoven content management software.
TeamSite comparison	A deployment configuration where two TeamSite *areas* are compared on the source host, and the differences are deployed to the specified target hosts. The source host must be configured as a TeamSite server.
test deployment	A deployment configuration that comes with OpenDeploy. Using this test determines whether your base server host is properly configured.
tier	A grouping of a source host and its target hosts, usually in the context of a multi-tiered deployment.
Tomcat server	Software included in the base server software which assists in the generation of the OpenDeploy user interface.
transactional deployment	A feature which restores one or more targets to their previous existing states in the event that the deployment is considered unsuccessful.
user interface	The browser-based graphical representation of selected OpenDeploy functions you can use as an alternative to modifying configuration files and entering commands at the command prompt. The user interface provides an easy way to perform OpenDeploy tasks and configurations.
User role	A lower level of OpenDeploy access. Users can only start those deployment configurations assigned to them by the administrator.
user translation	The switching of UNIX-based user ownership on a deployed file or directory between the source host and the target host.

verbose logging level	The highest level of deployment logging; detailed entries are written to the logs as the deployment occurs. This is the default logging level.
workarea	A TeamSite area where contributors keep their working files.

Index