

Deployit System Administration Manual

January, 2011

Contents

Preface	1
Installing Deployit	1
Prerequisites	1
Server Requirements	1
Unix Middleware Server Requirements	2
Windows Middleware Server Requirements	3
Client Requirements	3
Installation Procedure	3
Installing the Server	3
Deployit Directory Structure	3
Running the Setup Wizard	4
Changing the Admin Password	7
High Availability Setup	8
Configuring Deployit	8
Configuring Security	8
Security in Deployit	8
Granting, Revoking and Denying	9
Configuring Repository Security	9
Inherited Permissions	10
Permissions on Deployed Applications###	10
Security Configuration Example	10
Configuring LDAP Security	11
Configuring the Repository	12

Using a Database	12
Installing Plugins	12
Configuring Logging	13
Setting up Deployit	14
Starting and Stopping	14
Editing CIs	16
Maintaining Deployit	16
Creating Backups	16
Restoring Backups	16

Preface

This manual describes how to install and setup Deployit.

Installing Deployit

This section contains information on the installation of the Deployit server.

Prerequisites

Server Requirements

To install the Deployit server, the following prerequisites **must** be met:

- **Operating system:** Windows or Unix-family operating system running Java.
- **Java Runtime Environment:** JDK 1.6 (Sun/IBM or Apple)
- **RAM:** At least 2GB of RAM available for Deployit.
- **Harddisk space:** Sufficient harddisk space to store the Deployit repository. This depends on your usage of Deployit. See section **Determining Harddisk Space Requirements**.

Depending on the intended usage of Deployit, the following may also be required:

- **Database:** Deployit's Jackrabbit repository supports a number of different databases. For more information see Jackrabbit's persistence manager FAQ.
- **LDAP:** To enable group based security, an LDAP x.509 compliant registry is needed.

Determining Harddisk Space Requirements

The amount of disk space Deployit uses is dependent on how the product is used in your environment. It is most influenced by:

- the size and storage mechanism used for artifacts
- the number of packages in the system
- the number of deployments performed (more specifically, the amount of logging information stored)

To obtain an estimate, do the following:

- Install and configure Deployit for your environment as described in this document. Make sure you correctly set up the database-based or file-based repository.
- Estimate the number of packages to be imported (either the total number or the number per unit of time) (`NumPackages`)
- Estimate the number of deployments to be performed (either the total number or the number per unit of time) (`NumDeployments`)
- Record the amount of diskpace used by Deployit (`InitialSize`).
- Import a few packages using the GUI or CLI.
- Record the amount of displace used by Deployit (`SizeAfterImport`).
- Perform a few deployments.
- Record the amount of displace used by Deployit (`SizeAfterDeployments`).

The needed amount of diskpace in total is equal to:

$$\text{Space Needed} = ((\text{SizeAfterImport} - \text{InitialSize}) * \text{NumPackages}) + ((\text{SizeAfterDeployments} - \text{SizeAfterImport}) * \text{NumDeployments})$$

If `NumPackages` and `NumDeployments` are expressed per timeunit (e.g. the number of packages to be imported per month), then the end result represents the space needed per month as well.

Unix Middleware Server Requirements

Unix-based middleware servers that Deployit interacts with must meet the following requirements:

- **SSH Access:** The target systems should be accessible by SSH, i.e. they should run an SSH2 server.¹²
- **Credentials:** Deployit should be able to log in to the target systems using a login/password combination that allows it to perform at least the following Unix commands:
 - `cp`
 - `ls`
 - `mv`
 - `rm`
 - `mkdir`
 - `rmdir`

If the login user cannot perform these actions, a `sudo` user may be selected that can execute these commands.

¹The SSH daemon on AIX is known to hang with certain types of SSH traffic.

²For security, the SSH account that is used to access a host should have limited rights.

Windows Middleware Server Requirements

Windows-based middleware servers that Deployit interacts with must meet the following requirements:

- **CIFS/Telnet:** The target system should be accessible using Windows Telnet server running in *stream mode*.³
- **Directory shares:** The account used to access a target system should have access to the host's administrative shares such as **C\$**.

Client Requirements

The clients that access Deployit must meet the following requirements:

- **Web browser:** The following web browsers are supported:
 - IE 7.0 or up
 - Firefox 3.0 or up
 - Safari 3.0 or up
- **Flash Player:** A flash player is required, versions 9.0 and up are supported.

Installation Procedure

Installing the Server

Follow these steps to install the Deployit server application:

1. **Login to the server where Deployit will be installed.** The recommended way to install Deployit is to use a non-root user.
2. **Create an installation directory.**
3. **Copy the Deployit release distribution to the directory.**
4. **Unzip the release into the created directory.**

Deployit Directory Structure

Once the Deployit installation file is extracted, the following directory structure exists in the Deployit installation directory (in the remainder of the document this directory will be referred to as **DEPLOYIT_HOME**):

- **bin:** contains the server binaries
- **cli:** contains the client application, including binaries and libraries
- **doc:** contains the documentation
- **importablePackages:** default location for importable packages
- **lib:** contains server libraries
- **plugins:** contains the Deployit middleware plugins
- **recovery.dat:** stores tasks that are in progress for recovery purposes

³For security, the SSH account that is used to access a host should have limited rights.

Running the Setup Wizard

Run the Deployit Setup Wizard to start the Deployit server and make it ready for use. The command `deployit.sh -setup` starts the wizard. If you want to stop the Setup Wizard at any time, enter `exitsetup`. All changes to the configuration will be discarded.

The Setup Wizard displays the following welcome message:

```
Welcome to the Deployit setup.
You can always exit by typing 'exitsetup'.
To re-run this setup and make changes to the Deployit server configuration
you can run deployit.cmd -setup on Windows or deployit.sh -setup on Unix.
```

```
Do you want to use the simple setup?
Default values are used for all properties. To make changes to the default
properties, please answer no.
Options are yes or no.
[yes]:
```

Answer **yes** (or press Enter) to use the simple setup. Simple setup makes it easy to quickly get started with Deployit and to use the product's default configuration. See **Simple Setup** for more information.

Answer **no** to use the manual setup. Manual setup provides explicit control over all Deployit settings. See **Manual Setup** for more information.

Note: if you installed Deployit in the same location before, the Setup Wizard will display the following message:

```
An existing configuration was found. Do you want to edit it?
Options are yes or no. (selecting no will create an empty configuration)
[yes]:
```

Answer **yes** (or press Enter) to edit the existing configuration. The Setup Wizard will load all settings from the existing configuration and allow you to choose simple or manual setup.

Answer **no** to start over with an empty configuration.

Simple Setup

Using simple setup, the Setup Wizard will assume default values for all configuration parameters. Specifically, the following defaults will be used:

- The server will **not** use secure communication between the Deployit GUI and the Deployit server.
- The server will listen on Deployit's standard HTTP port (4516).
- The server will use a minimum of 3 and a maximum of 24 threads.
- The task recovery file will be deleted.
- Applications can be imported from the `importablePackages` directory.

The Setup Wizard will ask one more question:

Do you want Deployit to initialize the JCR repository?
Options are yes or no.
[yes]:

Answer **yes** (or press Enter) if you want the Deployit repository to be recreated. The Setup Wizard must have write access to the repository directory.

Answer **no** to leave the repository intact.

See **Finishing the Setup Wizard** for completing the setup process.

Warning: if you choose to recreate the Deployit repository and you have installed Deployit in the same location before, any information stored in the repository will be lost.

Manual Setup

The manual setup procedure contains the following steps:

Secure Communication Configuration

The Setup Wizard will show the following message:

Would you like to enable SSL?
Options are yes or no.
[yes]:

Answer **no** to use regular unsecured communication between the GUI and the server. Continue with the **port configuration** section.

Answer **yes** (or press Enter) if you want to use a secure connection from the GUI to the server.

If you answer **yes**, the Setup Wizard will ask the following question to help you configure secure communication:

Would you like Deployit to generate a keystore with a self-signed certificate for you?
N.B.: Self-signed certificates do not work correctly with some versions of the Flash Player and some browsers!
Options are yes or no.
[yes]:

Answer **yes** (or press Enter) if you want the Setup Wizard to generate a digital certificate automatically. The digital certificate is required to secure communication and is normally signed by a Certificate Authority (CA). The Setup Wizard can generate a *self-signed* certificate if there is no official certificate available. Beware that using a self-signed certificate may trigger security warnings in some Flash players and browsers. Continue with the **port configuration** section.

Answer **no** if you want to use your own keystore. Deployit uses the built-in Jetty webserver to communicate with the GUI. Jetty requires a certificate with the name **Jetty** to be present in the keystore.

The Setup Wizard prompts you for the following keystore information:

What is the path to the keystore?
[]:

What is the password to the keystore?

[] :

What is the password to the key in the keystore?

[] :

Enter the filesystem location of the keystore (for example, *mykeystore.jks*), the password to unlock the keystore and the password for the Jetty certificate in the keystore.

Port Configuration

The Setup Wizard shows the following question:

What http port number would you like the server to listen on?

[4516] :

Note: if you chose to enable secure communication, the default port will be *4517* instead of *4516*.

Enter the port number that the Deployit server listens on for connections.

Thread Configuration

The Setup Wizard shows the following questions:

Enter the minimum number of threads the HTTP server should use (recommended:

3 per client, so 3 for single user usage)

[3] :

Enter the minimum number of threads that the Deployit server uses to handle incoming connections. The recommended minimum number of threads is 3 per Deployit application client.

Enter the maximum number of threads the HTTP server should use (recommended :

3 per client, so 24 for 8 concurrent users)

[24] :

Enter the maximum number of threads that the Deployit server uses to handle incoming connections. The recommended maximum number of threads is 3 per Deployit application client.

Repository Configuration

The Setup Wizard shows the following questions:

Where would you like Deployit to store the JCR repository?

[repository] :

Enter the filesystem path to a directory where Deployit will create the repository. If the directory does not exist, the Setup Wizard will create it.

Do you want Deployit to initialize the JCR repository?

Options are yes or no.

[yes] :

Answer **yes** (or press Enter) if you want the Deployit repository to be recreated. The Setup Wizard must have write access to the repository directory.

Answer **no** to leave the repository intact.

Warning: if you choose to recreate the Deployit repository and you have installed Deployit in the same location before, any information stored in the repository will be lost.

Importable Packages Configuration

The Setup Wizard shows the following question:

```
Where would you like Deployit to import packages from?  
[importablePackages]:
```

Enter the filesystem path to a directory from which Deployit will import packages. The Setup Wizard assumes that this directory exists once the Deployit server starts and will not create it.

Finishing the Setup Process

Once you have completed configuration of the setup process, the Setup Wizard displays an overview of all selected options. The following text is an example:

```
Do you agree with the following settings for Deployit and would you like  
to save them?
```

```
Changes will be saved in deployit.conf
```

```
Security is always enabled
```

```
SSL will be disabled
```

```
HTTP port is 4516
```

```
HTTP server will use a minimum of 3 and a maximum of 24 threads
```

```
JCR repository home is at repository
```

```
JCR repository will be initialized.
```

```
Task recovery file will be deleted
```

```
Application import location is importablePackages
```

```
[yes]:
```

Answer **yes** (or press Enter) to store the configuration settings and end the Setup Wizard. If you selected the option to initialize the repository, this will be done now.

Answer **no** to abort the Setup Wizard.

If the Setup Wizard is successfully completed, it will display the following message:

```
You can now start your Deployit server by executing the command deployit.cmd  
on Windows or deployit.sh on Unix.
```

```
Note: If your Deployit server is running please restart it.
```

```
Finished setup.
```

Changing the Admin Password

By default, Deployit is installed with a special user with administrative permissions. This user has the username **admin** and password **admin**. As the last step in the installation, the admin password should be changed to something more secure. Issue the following commands to do this:


```
adminUser = security.read('admin')
adminUser.password = 'newpassword'
security.modifyUser(devUser)

# Test whether the change is successful
security.logout()
security.login('admin', 'newpassword')
```

High Availability Setup

Deployit can be configured to ensure maximum uptime of the application. In such a high availability setup, two instances of Deployit are running in an *active/passive* configuration. At any one time, only one Deployit instance is active but as soon as a failure is detected, the passive Deployit instance is activated and the failed instance is taken down for repair.

The easiest way to achieve such a configuration is by using the same configuration for each Deployit instance.

Warning: unpredictable results may occur when running two Deployit instances against the same repository *at the same time*.

When switching from one Deployit instance to another, any running tasks will be recovered by the new instance (see **Task Recovery**).

To configure such a setup, a router with *active/passive* support must be used.

Configuring Deployit

This section contains information on the configuration of the Deployit server.

Configuring Security

Security in Deployit

Deployit supports a fine-grained access control scheme to ensure the security of your middleware and deployments. Deployit's security mechanism is based on the concepts of *principals*, *permissions* and *privileges*.

Principals

A (security) principal is an entity that can be authenticated and that can be assigned rights over resources in Deployit. Out of the box, Deployit supports only users as principals — users are authenticated by means of a username and password and rights within Deployit are assigned to the user itself. When using an **LDAP** repository, groups in LDAP are also treated as principals.

There is one special user, **admin**, who has special rights in Deployit. This user is allowed to grant and revoke security permissions.

Permissions

Permissions in Deployit are rights to execute particular actions on certain CIs. In addition, the permissions also allow the user to make changes to the repository that are necessary to carry out the granted actions. If granting the permission without specifying the target CI(s), broad repository access is allowed. See the list below.

Deployit supports the following permissions:

- **read**. The right to see particular CIs.
- **login**. The right to log into the Deployit application. The user has no access to nodes in the repository.
- **import#initial**. The right to import a package for which the application does not yet exist in the repository and for which a new application will be created. This implies read and write access to the **Applications** node. The permission can also be given for specific *Application* CIs.
- **import#upgrade**. The right to import a package for which the application already exists in the repository. This implies read and write access to the **Applications** node. The permission can also be given for specific *Application* CIs.
- **deploy#initial**. The right to perform an initial deployment of a package to an environment. This implies read and write access to the **Environments** and **Infrastructure** nodes. The permission can also be given for specific *Environment* CIs.
- **deploy#upgrade**. The right to perform an upgrade of a package on an environment. This implies read and write access to the **Environments** and **Infrastructure** nodes. The permission can also be given for specific *Environment* CIs.
- **discovery**. The right to perform discovery. This implies read and write access to the **Infrastructure** node.
- **repo#edit**. The right to edit (create and modify) CIs. This implies write access to the **Applications**, **Environments** and **Infrastructure** nodes. The user must also have read access to CIs to be able to edit them. The permission can also be given for specific CIs.
- **undeploy**. The right to undeploy an application. This implies read and write access to the **Environments** and **Infrastructure** nodes. The permission can also be given for specific *Environment* CIs.

Granting, Revoking and Denying

Access rights in Deployit can be *granted* to a principal or *revoked* from a principal. When rights are granted, the principal is allowed to perform some action or access repository entities. Rights once granted can be revoked again to prevent the action in the future. Rights can also be explicitly *denied*. Denying permission acts as a negative grant — the right is explicitly disallowed.

Access rights can be used stand-alone or in combination with one or more CIs. In the former case, the principal will have access to **all** CIs associated with the permission. In the latter case, the access rights will be restricted to the particular CIs. For example, granting *import#initial* to a principal allows the principal to import any application. Granting *import#initial* to *Applications/PetClinic* allows the principal to import only *PetClinic* packages.

Configuring Repository Security

Security in the Deployit repository can be configured using the **Command Line Interface**. See the **Deployit Command Line Manual** for more information.

Creating Users

Deployit can only create users in its own repository, even if it is configured to use an **LDAP** repository for authentication and authorization. To do this, use a statement such as the following:

```
deployer = security.createUser("john", "secret")
```

Granting Permissions

To grant a particular permission to a principal, use a statement such as the following:

```
security.grant("import#initial", "john")
```

To grant a particular permission to a principal on a resource, use a statement such as the following:

```
security.grant("read", "mary", ['Environments/Dev'])
```

Revoking Permissions and Privileges

To revoke a particular permission from a principal, use a statement such as the following:

```
security.revoke("read", "john", ['Environments/Dev'])
```

Inherited Permissions

Permissions in Deployit are *inherited* for all CIs that are contained in the CI node you specified the permissions for. For example, if you have read permission on *Environments/Dev*, you will also have read permission on all of the CIs under this node such as deployed applications. This is what happens when you use stand-alone permissions. If you use permissions on specific CIs, these permissions are set only on the specified CIs and are inherited from this CI onwards.

Permissions on Deployed Applications###

Giving users permissions on deployed applications requires some knowledge of how these CIs are stored in the repository. As described in the **Deployit Reference Manual**, deployed applications are stored under both the **/Environments** as well as **/Infrastructure** nodes. Giving users read permission on a deployed application involves giving them permissions under both nodes.

Security Configuration Example

Let's illustrate the security setup with an example.

In a typical medium to large size company, there several different groups of people that perform tasks related to deployments. There are administrators that install, test and maintain hardware, there are deployers that deploy applications to development, test, acceptance and production environments. And finally there are the developer who build these applications.

Translating this into Deployit terms:

- **administrators:** permission to create, update and delete infrastructure as well as permission to create, update and delete environments. (all handled by `_infrastructure#management` permission)
- **deployers:** permission to import new applications (*import#initial*), deploy to DEV, TEST and view PROD (*deploy#initial* and *deploy#upgrade* on environments DEV and TEST, *read* rights on environment PROD).
- **senior deployers:** permission to import new applications (*import#initial*), deploy to DEV, TEST and PROD (*deploy#initial* and *deploy#upgrade* on environments DEV, TEST and PROD).

- **developers:** permission to import new versions of existing applications (*import#upgrade*) and to upgrade existing deployments (*deploy#upgrade*).

See the **Deployit Command Line Interface (CLI) Manual** for the exact commands to implement this.

Configuring LDAP Security

By default, Deployit authenticates users and retrieves authorization information from its repository. Deployit can also be configured to use an LDAP repository to authenticate users and to retrieve role (group) membership. In this scenario, Deployit works with both users and groups as principals. This means that rights can be assigned to both users and groups. The rights assigned to a principal are always stored in the JCR repository.

Deployit treats the LDAP repository as **read-only**. This means that Deployit will use the information from the LDAP repository, but can not make changes to that information.

When authenticating a user, Deployit first tries to locate the user in the LDAP repository. If this fails, Deployit will check its own repository as a backup.

To configure Deployit to use an LDAP repository, the built-in JCR repository, Jackrabbit, must defer to the LDAP server for authentication. This requires modification of the default Deployit Jackrabbit configuration. Follow these steps:

1. **Create conf/jackrabbit-jaas.config file.** The file must be accessible for the Deployit server. The following is a sample file:

```
Jackrabbit {
    org.apache.jackrabbit.core.security.authentication.DefaultLoginModule
        sufficient adminId=admin anonymousId=anonymous;
    com.xebialabs.deployit.security.LdapLoginModule required
        userProvider="ldap://localhost:14516/ou=system"
        userFilter="(&(uid={USERNAME}))(objectClass=inetOrgPerson)"
        useSSL=false
    principalProvider=com.xebialabs.deployit.security.LdapPrincipalProvider;
};
```

Modify the LDAP settings (URL, port or user filter, etc.) to suit your needs.

2. **Modify the Deployit server startup command.** Notify the Deployit server of the new configuration file by including the following parameter on the Java command line:

```
-Djava.security.auth.login.config="$DEPLOYIT_HOME/conf/jackrabbit-jaas.config"
```

3. **Extract the jackrabbit-repository.xml from the jar, and change it**

```
cd $DEPLOYIT_HOME/conf
jar xvf ../lib/server-core-3.1.jar jackrabbit-repository.xml
```

Change the jackrabbit-repository.xml by looking up the Security-block, and changing it to look like this:

```

<Security appName="Jackrabbit">
  <SecurityManager class="org.apache.jackrabbit.core.DefaultSecurityManager" workspaceName="secun
  <AccessManager class="org.apache.jackrabbit.core.security.DefaultAccessManager" />
</Security>

```

The class **LdapPrincipalProvider** is an example of how to connect Deployit to LDAP through Jackrabbit. If your LDAP server has a different structure or to connect Jackrabbit to another user store, see the information about Jackrabbit configuration on the the Jackrabbit website.

Configuring the Repository

Using a Database

Deployit can also use a database to store it's repository in. The built-in Jackrabit repository must be configured to make this possible. The two relevant components are Jackrabbit's **PersistenceManager** (used for storing nodes and revisions) and **DataStore** (optionally used for storing large binary objects). In the default Deployit configuration, the Derby database and file system DataStore are used.

This is an example of how to configure Deployit to use a database.

First, extract the Jackrabbit configuration file and copy it to the `conf` directory where it will be picked up by Deployit:

```

cd $DEPLOYIT_HOME/conf
jar xvf ../bin/deployit-3.0-server.jar jackrabbit-repository.xml

```

Edit the Jackrabbit configuration file and change each **PersistenceManager** to the following:

MySQL ⁴.

```

<PersistenceManager class="org.apache.jackrabbit.core.persistence.pool.MySqlPersistenceManager">
  <param name="url" value="jdbc:mysql://localhost:3306/deployit" />
  <param name="user" value="deployit" />
  <param name="password" value="deployit" />
  <param name="schemaObjectPrefix" value="${wsp.name}_" />
</PersistenceManager>

```

DB2

Oracle

For more information about using a database with Jackrabbit, see it's **PersistenceManager** FAQ and **DataStore** FAQ.

Installing Plugins

Deployit uses plugins to communicate with different types of middleware. The Deployit server dynamically loads plugins when it starts. Any plugins added or removed when Deployit server is running will not take effect until the server is restarted.

⁴The MySQL database is not suited for storage of large binary objects, see the MySQL bug tracker.

Plugins are stored in the `plugins` directory in the Deployit installation directory.

Installing a Plugin

To install a new plugin, simply stop the Deployit server and copy the plugin JAR archive into the `plugins` directory, then restart the Deployit server.

Deinstalling a Plugin

To deinstall a plugin, simply stop the Deployit server and remove the plugin JAR archive from the `plugins` directory, then restart the Deployit server.

Configuring Logging

Out of the box, Deployit server writes informational, warning and error log messages to standard output when running. It is possible to change this behavior to write log output to a file or to log output from a specific source.

Deployit uses the Logback logging framework for its logging. To change its behavior, edit the file `logback.xml` in the `conf` directory of the Deployit server installation directory.

The following is an example `logback.xml` file:

```
<configuration>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <!-- encoders are assigned the type
         ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->
    <encoder>
      <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
    </encoder>
  </appender>

  <!-- Create a file appender that writes log messages to a file -->
  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <layout class="ch.qos.logback.classic.PatternLayout">
      <pattern>%-4relative [%thread] %-5level %class - %msg%n</pattern>
    </layout>
    <File>log/my.log</File>
  </appender>

  <!-- Set logging of classes in com.xebialabs to DEBUG level -->
  <logger name="com.xebialabs" level="debug"/>

  <!-- Set logging of class HttpClient to DEBUG level -->
  <logger name="HttpClient" level="debug"/>

  <!-- Set the logging of all other classes to INFO -->
  <root level="info">
    <!-- Write logging to STDOUT and FILE appenders -->
    <appender-ref ref="STDOUT" />
    <appender-ref ref="FILE" />
  </root>
</configuration>
```

Place a file called `logback-test.xml` on the server classpath to temporarily change the server's logging settings.

For more information see the Logback website.

Setting up Deployit

This section describes how to setup Deployit server in your environment.

Deployit must be setup for your environment before it can be used to execute deployments. This entails the following steps:

1. **Start the Deployit server.**
2. **Discover your middleware.** Deployit can inspect your environment and automatically create CIs for your middleware. Alternatively, you can use a bulk-import to import your middleware or create them by hand.
3. **Add the discovered middleware to an environment.** CIs must be grouped in an environment to use them for deployment.

Setup of Deployit is performed using the Deployit **Command Line Interface (CLI)**. For more information about the CLI, see the **Deployit Command Line Interface (CLI) Manual**.

Starting and Stopping

Starting the Server

Open a terminal window and change to the `DEPLOYIT_HOME` directory. Start the Deployit server with the command:

```
bin/server.sh
```

on Unix and

```
bin/server.cmd
```

on Windows.

By starting the server with the `-h` flag, a message is printed that shows the possible options it supports:

```
java -cp deployit-server-<version>.jar [options...] com.xebialabs.deployit.Deployit arguments...
  -reinitialize : Reinitialize the repository, only useful with -setup
  -setup       : (Re-)run the setup for Deployit
```

The command line options are:

- **-reinitialize** — tells Deployit to reinitialize the repository. Used only in conjunction with **-setup**.
- **-setup** — runs the Deployit Setup Wizard.

- **-test-mode** — enables Deployit server test-mode. When test-mode is enabled, WebDAV access to the JCR repository is installed. The repository can be accessed using a URL such as `http://localhost:4516/repository/`

Starting Deployit in the Background

By running the `server.sh` or `server.cmd` command, the Deployit server is started in the foreground. To run the server as a background process, use:

```
nohup bin/server.sh &
```

on Unix or run Deployit as a service on Windows.

Java Properties

Deployit server also responds to certain Java properties that influence its behavior. These properties may be set in the environment (e.g. by executing `export jetty.host=127.0.0.1` in the terminal used to start the server) or by passing them to Java on the command line (for instance, by adding the flag `-Djetty.host=127.0.0.1` to the command that starts the server). The following options are supported:

- `deployit.supported.extensions.in.archive` — sets the file extensions for which Deployit will attempt to replace placeholders. See the **Deployit Packaging Manual** for more information.
- `jetty.host` — sets the host that Deployit's embedded HTTP server binds to. Default value is `localhost`.
- `ssh.privatekey.filename` — supplies Deployit with a private key file that is used to access a remote host using public-private key security over SSH.
- `ssh.privatekey.passphrase` — supplies Deployit with the passphrase to use in conjunction with the SSH private key.
- `com.xebia.ad.donotcleantemporaryfiles` — **debugging only**: tells Deployit to retain the temporary files it uses to execute commands on a remote host.
- `com.xebialabs.deployit.jetty.ClassPathResourceContentServlet.fallbackDirectory` — **debugging only**: tells Deployit to serve classpath resources from a different directory.
- `com.xebialabs.deployit.plugin.was.disableDaemon` — **debugging only**: tells Deployit **not** to use the daemon to speed up command execution on WAS.

Note: Deployit's main configuration takes place in the `deployit.conf` file. In an effort to consolidate configuration in one place, the properties mentioned above are expected to be migrated to the `deployit.conf` file in a future release.

Stopping the Server

It is possible to stop the Deployit server using a REST API call. The following is an example of a command to generate such a call (replace `admin:admin` with your own credentials):

```
curl -X POST --basic -u admin:admin
http://admin:admin@localhost:4516/deployit/server/shutdown
```

This requires the external `curl` command, available for both Unix and Windows.

Editing CIs

The CIs in the Deployit repository can also be edited, either using the command line interface (CLI) or graphical user interface (GUI). See the respective manuals for more details.

Maintaining Deployit

This section describes how to maintain the Deployit server in your environment.

Creating Backups

To create a backup of Deployit, several components may need to be backed up depending on your configuration:

- **JCR repository.**
 - Built-in repository: Create a backup of the built-in JCR repository by backing up the files in the repository directory.
 - Database repository: Create a backup of the database using the tools provided by your vendor.
- **Configuration.** Create a backup of the Deployit configuration by backing up the files in the `conf` directory in the installation directory.

Note: Deployit **must not** be running when you are making a backup. Schedule backups outside of regular working hours to ensure the server is not being used.

Restoring Backups

To restore a backup of Deployit, restore one of the following components:

- **JCR repository.**
 - Built-in repository: Remove the repository directory and replace it with the backup.
 - Database repository: Restore a backup of the database using the tools provided by your vendor.
- **Configuration.** Remove the `conf` directory in the `DEPLOYIT_HOME` directory and replace it with the backup.

Note: Deployit **must not** be running when you are restoring a backup.