

Deployit Reference Manual

December, 2010

Contents

| | |
|---|----------|
| Preface | 1 |
| Deployment Overview | 1 |
| Deployit Concepts | 2 |
| Archetypes | 2 |
| Artifacts | 2 |
| Command Line Interface (CLI) | 2 |
| Configuration Items (CIs) | 2 |
| Deployed Items | 3 |
| Deploying an Application | 3 |
| Deployment ARchive (DAR) Format | 4 |
| Deployment Package | 4 |
| Environment | 4 |
| Middleware Resources | 4 |
| Plug-in | 4 |
| Repository | 5 |
| Containment and References | 5 |
| Deployed Applications | 5 |
| Runbook | 5 |
| Step | 6 |
| Steplist | 6 |
| Task | 6 |
| Task Recovery | 7 |
| Task Reloading | 7 |

| | |
|--|----------|
| Task State | 7 |
| Undeploying an Application | 7 |
| Unified Deployment Model (UDM) | 7 |
| Upgrading an Application | 8 |
| Configuration Item Reference | 8 |

Preface

This manual contains general background information for users of Deployit.

Deployment Overview

Deployit is the first out-of-the-box deployment automation solution that allows non-experts to perform application deployments. A deployment consists of all the actions needed to install, configure and start an application on a target environment. Deployments are defined by:

- A **Package** or *what* is to be deployed.
- A **Environment** or *where* the package is to be deployed.
- A **Deployment** or *customizations* to the package to be deployed.

Once the *what* and the *where* are known, h Deployit (and it's runbooks) takes care of the *how* by putting together a list of steps (such as copying files, starting / stopping application servers) that perform the actual deployment.

Deployments can be categorized into two major categories:

- **Initial deployments**
- **Upgrade deployments**

A deployment is an *initial* deployment when the package is deployed to the environment for the first time. This type of deployment typically requires some effort to get right because all deployed items must be specified and configured. A user familiar with the middleware setup and with experience in this area is normally involved in this activity.

Once the initial deployment has been configured, all subsequent deployments of the package to the environment are *upgrade* deployments. These deployments reuse the deployed items from the initial deployment and can typically be performed without any further configuration. Consequently, users without any knowledge of the middleware setup can perform these deployments.

Deployit Concepts

These are the main concepts in Deployit, in alphabetical order.

Archetypes

An *archetype* is a die for creating many similar **Configuration Items (CIs)** and are a way to reduce duplication of data. An archetype contains default values for properties that are the same for all of the CIs that are created based on the archetype. CIs created from an archetype have the option of overriding the values from the archetype. A CI created from an archetype keeps the link to the archetype so that updates to the archetype's properties are immediately made for all of the related CIs. Archetypes can also be nested.

For example, if we define an archetype **UnixHost** that has the property *OS Family* set to the value *UNIX*. If we create host CIs **web1** and **web2** based on the **UnixHost** archetype, both host CIs inherit the *OS Family* property from the archetype. If we change the *OS Family* property to *WINDOWS* on the archetype, the value will automatically change for both host CIs. If we change the *OS Family* property to *WINDOWS* on **web1**, this overrides the value on the **UnixHost** archetype. Any changes to *OS Family* on the archetype now only affect the **web2** CI.

Artifacts

Artifacts are files containing application resources such as code or images. The following are examples of artifacts:

- a WAR file.
- an EAR file.
- a file containing static content such as HTML or images.
- a folder containing SQL upgrade files.

In the **Unified Deployment Model**, a complete application (consisting of both artifacts and middleware resources) is contained in a single deployment package.

Command Line Interface (CLI)

The Deployit CLI provides a way to programmatically interact with Deployit. The CLI can be programmed using the **Python** programming language. For more detailed information, see the **Deployit Command Line Manual**.

Configuration Items (CIs)

A *configuration item* (CI) is a fundamental structural unit in a configuration management system. In Deployit, a CI is a container for properties and relations to other CIs. A CI holds information about a part of the infrastructure, application or deployment. A CI has a certain *type* that determines what information it contains and what it can be used for.

The following are some examples of CIs:

- **web1.acme.com**. This is a CI of type *Host*.
- **Oracle 11g on db1.acme.com**. This is a CI of type *Database*. Note that this CI links to a *Host* CI indicating where the database is running.

- **Tomcat 7.0 on web1.acme.com.** This is a CI of type *TomcatServer*. Note that this CI links to a *Host* CI indicating where the server is running.
- **PetClinic.** This is a CI of type *Application*.
- **PetClinic/1.0.** This is a CI of type *DeploymentPackage*. Note that this CI links to an *Application* CI indicating which application the package contains.
- **Test.** This is a CI of type *Environment*. It contains CIs to which applications can be deployed.
- **PetClinic/1.0 on Test.** This is a CI of type *Deployment*. Note that this CI links to an *Application* CI indicating the application that is deployed, and an *Environment* CI indicating where the application is deployed.

Deployit CIs all share certain properties:

- **id.** This is the id of the CI, a unique identifier for this CI that can be used to reference it. The id determines the place of the CI in the **Repository**.
- **archetypeId.** This specifies which archetype (if any) the CI has. See section **Archetypes** for more information.

Some properties of a CI are mandatory (must be given a value before the CI can be used), others are optional. To determine which properties are available and which are mandatory or optional for a CI, see the **CI Reference** or use the help facility in the Deployit **CLI**.

Deployed Items

Deployed items are members of a **deployment package** that have been deployed on some target middleware. The deployed item can be configured for the specific target it is mapped to during the deployment. When configuring a deployed item it is possible to override any CI properties that the CI has in the generic package. This allows a single package to be used in different environments.

Deploying an Application

This process installs a particular application (represented by a **deployment package**) on an environment. Deployit copies all necessary files and makes all configuration changes to the target middleware that are necessary for the application to run. If there is already a version of the application running on the environment, the deployment is an *upgrade* rather than an initial deployment (see **Upgrading an Application**). Before performing an initial deployment, all deployed items must be configured correctly for the deployment to work.

Deployment ARchive (DAR) Format

The DAR format is the native format Deployit supports for **deployment packages**. A DAR file is basically a Java **jar** archive containing a specially formatted manifest. The archive contains the package's **artifacts** while the **middleware resources** are described in the archive's manifest file. Deployit processes the manifest file to know which CIs are stored in the archive.

For a comprehensive description of the manifest format, see the **Deployit Packaging Manual**.

Deployment Package

A deployment package is an archive containing a particular version of an application, containing all **artifacts** and **middleware resources** that the application needs. It is independent of the environment it is deployed in (see **Unified Deployment Model**).

Deployit accepts packages in the **Deployment ARchive (DAR)** format.

Environment

An **Environment** is a grouping of infrastructure items, such as hosts, servers, clusters, etc. Environments can contain any combination of infrastructure items that are used in your situation. An environment is used as the target of a deployment, allowing members of the **deployment package** to be mapped to members of the environment.

Middleware Resources

Middleware resources are pieces of middleware configuration that an application is dependent on. The following are examples of middleware resources:

- a queue.
- a topic.
- a datasource.
- a classpath entry.

In the **Unified Deployment Model**, a complete application (consisting of both middleware resources and artifacts) is contained in a single deployment package.

Plug-in

A plug-in is a self-contained piece of functionality that adds capabilities to the Deployit system. Plugins contain **Runbooks**, **Steps** and **Configuration Item** types.

Repository

The *repository* is a storage space for all things Deployit knows about. This includes configuration items (CIs), binary files (such as **deployment packages**) and Deployit's security configuration (user accounts and rights). The repository can be stored on disk (default) or in a database (see **Configuring Database Storage**).

In Deployit, the repository conforms to the Java Content Repository (JCR) standard. Deployit's repository has a hierarchical layout and a version history. The repository stores all CIs of all types. The top-level folders indicate the type of CI stored below it. Depending on the type of CI, the repository stores it under a particular folder:

- *Application* CIs are stored under the **/Applications** folder.

- *Environment* CIs are stored under the **/Environments** folder.
- Middleware CIs (*Host*, *Server*, etc.) are stored under the **/Infrastructure** folder.
- *Archetypes* are stored under the **/Archetypes** folder.

Containment and References

Deployit's repository contains CIs that are containers for other CIs. There are two ways in which CIs can refer to each other:

- **Containment.** In this case, one CI contains another CI. If the parent CI is removed, so is the child CI. An example of this type of reference is an *Environment* CI and its deployed applications.
- **Reference.** In this case, one CI refers to another CI. If the referring CI is removed, the referred CI is unchanged. Removing a CI when it is still being referred to is not allowed. An example of this type of reference is an *Environment* CI and its middleware. The middleware exists in the **/Infrastructure** folder independently of the environments the middleware is in.

Deployed Applications

Deployed applications have a special structure in the repository. A deployed application is the result of deploying a *package* to an *environment*. While performing the deployment, package members are installed as *deployed items* on individual environment members. In the repository, the *deployed application* CI is stored under the *Environment* node. Each of the *deployed items* are stored under the infrastructure members in the *Infrastructure* node.

So, deployed applications exist in both the **/Environment** as well as **/Infrastructure** folder. This has some consequences for the security setup. See the **Deployit System Administration Manual** for details.

Runbook

A runbook is a component that generates a **steplist** when presented with a **task**. A runbook can produce **steps** for any task or only for particular tasks. A runbook is integrated into Deployit by means of a **plug-in**.

Deployit ships with the following runbooks:

- **Tomcat runbook.** This runbook makes it possible to deploy to the **Tomcat** application server.
- **JBoss runbook.** This runbook makes it possible to deploy to the **JBossAS** application server.

XebiaLabs also provides the following optional runbooks:

- **WAS runbook.** This runbook makes it possible to deploy to the **WebSphere Application Server (WAS)**.
- **WLS runbook.** This runbook makes it possible to deploy to the **WebLogic** application server.
- **IBM MQ runbook.** This runbook makes it possible to deploy to the **IBM MQ Series** platform.
- **IBM Portal runbook.** This runbook makes it possible to deploy to the **IBM WebSphere Portal** application server.

It is also possible to customize Deployit by building your own runbook from scratch or by changing an existing runbook. Please see the **Deployit Programmers Manual** for more information.

Step

A **step** is one concrete action to be performed to accomplish a **task**. A step is created by a **runbook** and contained in a **steplist**. Deployit ships with a number of standard infrastructure steps. Other middleware-specific steps are contributed by the plugins.

The following are examples of steps:

- copy file /foo/bar to host1, directory /bar/baz.
- restart the Apache HTTP server on web1.
- create a virtual host in the WAS server on was1.
- run the SQL installation files on b1.

Steplist

A **steplist** is a sequential list of **steps** that is generated by one or more runbooks when a task is executed. The **steplist** contains all steps that Deployit will execute to perform the task.

Task

A **task** is an activity in Deployit. When starting a deployment, Deployit will create and start a task. The task contains a list of **steps** that must be executed to successfully complete the task. To execute the task, Deployit will execute each of the steps in turn. When all of the steps are successfully executed, the task itself is successfully executed. If one of the steps fails, the task itself is marked stopped.

Deployit supports the following tasks:

- **deploy application**. This task deploys a package onto an environment.
- **upgrade application**. This task upgrades an existing deployment of an application.
- **undeploy application**. This task undeploys a package from an environment.
- **discovery**. This task discovers middleware on a host.

Task Recovery

Deployit periodically stores a snapshot of the tasks in the system to be able to recover tasks if the server crashes. If this happens, Deployit reloads the tasks from the recovery file when it restarts. The tasks, deployed item configuration and generated steps will all be recovered. Tasks that were running in Deployit when the server crashed will be put in **STOPPED** state so the user can decide whether to rerun it or cancel it. Tasks that were not yet started (for instance, a deployment for which only mappings have been configured, but no **steplist** was generated) will not be recovered.

Task Reloading

When a user logs in, the Deployit GUI loads any tasks that are pending for this user from the repository and displays them in tabs. This includes tasks that were stopped or executing when the user closed his browser,

Task State

Deployit allows a user to interact with the task. In addition to starting a task, a user can:

- **stop the task.** Deployit will wait for the currently executing step to finish and will then cleanly stop the task. Note that, due to the nature of some steps, this is not always possible. For example, a step that calls an external script may hang indefinitely.
- **abort the task.** Deployit will attempt to kill the currently executing step. If successful, the aborted step and task are marked failed.
- **cancel the task.** Deployit will remove the task from the system. If the task was executing before, the task will be archived since it may have made changes to the middleware. If the task was pending and never started, it will be removed.

Undeploying an Application

This process removes an application from an environment. Deployit stops the application and removes the package from the target middleware.

Unified Deployment Model (UDM)

The UDM is XebiaLabs' model for describing deployments and is used in Deployit. The UDM consists of the following components:

- **Deployment Package.** This is an environment-independent package containing a complete application. The application contains both **artifacts** as well as associated **middleware resources**.
- **Environment.** This is an environment containing deployment targets, i.e. resources that applications can be deployed on. An example is a test environment containing a cluster of WebSphere servers.
- **Deployment.** This is the process of installing and configuring a *deployment package* onto a specific *environment*.

Upgrading an Application

This process replaces an application deployed to an environment with another version of the same application. If the application is not running on the environment, the deployment is an *initial* deployment (see **Deploying an Application**). When performing an upgrade, most deployed items can be inherited from the initial deployment. Deployit recognizes which artifacts in the deployment package have changed and deploys only those artifacts.

Configuration Item Reference

Deployit ships with a number of **configuration items** that you can use to define your middleware configuration and deploy applications. This section contains an overview of these CIs, their purpose and the properties they support. This section of the manual was generated from the Deployit **CLI** interface.