

Deployit Upgrade Manual

Version 3.9.1

Table of Contents

Table of Contents	2
The upgrade process	3
Upgrade notes	3
Performing the upgrade	3
Upgrading the Server	3
Upgrading the CLI	4
Specific upgrade notes	4
Upgrading to Deployit 3.9	4
Upgrading plugins	4
Checksum property	4
Importing (old style manifests)	4
Upgrading to Deployit 3.8.4	4
Upgrading to Deployit 3.8	5
Repository Upgrade	5
Release Dashboard	5
Placeholder scanning	5
Custom Java plugins	5
Upgrading to Deployit 3.7	5
Security upgrade	5
Additional upgrade steps	7
Completing the migration	7
Upgrading to Deployit 3.6	7
Upgrading from Deployit 3.0 or earlier	7

The upgrade process

Overall, the upgrade process consists of the following steps:

- Obtain a new version of the Deployit software (the main product and/or plugins) from XebiaLabs.
- Read the new version's **release notes** so you are aware of the new functionality and possible upgrade considerations.
- Read the new version's **upgrade manual** (this document) so you are aware of possible upgrade considerations.
- Stop the current version of Deployit if it is still running, making sure there are no running tasks active.
- Create a brand-new installation directory for the new version of Deployit so the existing version is still available in case of problems.
- Extract the new Deployit software release into the new installation directory.
- Copy the data from the previous Deployit installation directory into the new installation directory.
- Start the new version of Deployit.

See the section **Performing the Upgrade** below for a detailed explanation of these steps.

Upgrade notes

- It is possible to skip Deployit versions when upgrading. Deployit will sequentially apply any upgrades for the intermediate (skipped) versions. **Please read the specific upgrade instructions for each of the versions carefully.**
- The new version of Deployit may not be compatible with the existing version of your plugins. If this is the case, you'll need to download an updated version of your plugin as well. **Please read the specific upgrade instructions for each of the versions carefully.**
- If a repository upgrade is required, Deployit will detect that it is running against an old repository and will automatically execute an upgrade when it is first started. The server log will contain extensive logging of the repository upgrade process.
- Plugin versions are related to the version of Deployit that they are compatible with. For instance, WAS plugin version 3.6.0 requires **at least** Deployit server version 3.6.0. This version of the WAS plugin should also work in Deployit 3.7.0 unless stated otherwise in this document.

Performing the upgrade

To begin upgrading Deployit, first unpack the distribution archive. The distribution archive contains the following:

- A server archive.
- A CLI archive.

The release notes are stored in the server's **doc** directory. Please read them carefully before performing the upgrade.

Upgrading the Server

To upgrade an existing Deployit server installation, do the following:

1. Create a directory for the new Deployit server installation, including the new Deployit server version number in the directory name.
2. Extract the server archive in this directory.
3. Copy the contents of the **conf** directory from the previous installation into the new installation directory.
4. Copy the entire **repository** directory from the previous installation into the new installation directory.
5. Copy the contents of the **importablePackages** directory from the previous installation into the new installation directory.
6. Unless new versions of your plugins were provided with the new Deployit version, copy the contents of the **plugins** directory from the previous installation into the new installation directory.
7. Copy the contents of the **ext** directory from the previous installation into the new

installation directory.

8. **DO NOT** copy the contents of the **hotfix** directory (unless instructed) because hotfixes are version specific.
9. If you added libraries to Deployit's **lib** directory (for instance, database drivers), copy the additional libraries from the previous installation into the new installation directory.
10. If you have made any changes to the Deployit server startup scripts (*server.sh* or *server.cmd*), manually re-do these changes in the new installation directory.

Note: please make sure that the plugins and extensions in the old Deployit installation are compatible with the new Deployit server version.

This completes upgrading of the Deployit server.

Upgrading the CLI

To upgrade an existing Deployit CLI installation, do the following:

1. Create a directory for the new Deployit CLI installation, including the new Deployit CLI version number in the directory name.
2. Extract the CLI archive in this directory.
3. Copy the contents of the **plugins**, **ext** and **hotfix** directories from the previous installation into the new installation directory.
4. If you have made any changes to the Deployit CLI startup scripts (*cli.sh* or *cli.cmd*), copy these from the **bin** directory in the previous installation into the new installation directory.

This completes upgrading of the Deployit CLI.

Specific upgrade notes

This section describes specific considerations for migrating from or to a particular Deployit version:

- [Upgrading to Deployit 3.9](#)
- [Upgrading to Deployit 3.8.4](#)
- [Upgrading to Deployit 3.8](#)
- [Upgrading to Deployit 3.7](#)
- [Upgrading to Deployit 3.6](#)
- [Upgrading from Deployit 3.0 or earlier](#)

Upgrading to Deployit 3.9

Upgrading plugins

Plugins from Deployit version 3.8 will work in Deployit 3.9. There is no need to upgrade plugins when you are upgrading to Deployit 3.9.

Checksum property

In Deployit 3.9, a new system property called `checksum` has been introduced on configuration items of type `udm.BaseDeployableArtifact`. This property influences how Deployit will calculate upgrades of deployments (See: [Packaging manual](#)). If you've previously defined a property called `checksum` on your configuration items, please rename it to avoid clashes with the newly introduced `checksum` property.

Importing (old style manifests)

On importing it is now possible to pass in a `CI-Name` for non-artifact configuration items. The behaviour for both Artifacts and non-Artifacts is now the same, both will be stored under the `CI-Name` value if it is given, and fallback to the Manifest entry name `Name`.

Upgrading to Deployit 3.8.4

- **DEPLOYITPB-4188** - In versions of Deployit before 3.8.4, the `envVars` property on the `generic.Container` type of the `generic-plugin` (and therefore also on the `sql.SqlClient` type of the `database-plugin` and its subtypes) was resolved using FreeMarker. From Deployit 3.8.4 onwards this is no longer done as their values should refer to Unix or Windows environment variables which will be resolved by the shell.

Upgrading to Deployit 3.8

Repository Upgrade

Deployit 3.8 has an updated repository format. Data from a pre-3.8 repository will automatically be converted by the Deployit Server.

Release Dashboard

The Release Dashboard feature has changed to make it easy to reuse a deployment pipeline for many applications. A new CI has been introduced, *release.DeploymentPipeline* which is stored under the new *Configuration* root node. Each application that should be used in the Release Dashboard can link to the required deployment pipeline.

When starting Deployit 3.8 for the first time, any deployment pipelines configured in 3.7 will be converted to 3.8 deployment pipeline CIs.

Placeholder scanning

Placeholder scanning in archives (EAR, WAR, etc.) is now **disabled** by default. To enable it, edit `deployit-defaults.properties` and add the following line:

```
udm.BaseDeployableArchiveArtifact.scanPlaceholders=true
```

Custom Java plugins

If you have custom Java plugins written against the Deployit 3.7 API, you may need to make some changes to take advantage of the new features in Deployit 3.8.

Deployit 3.8 has changed the API to manage deployments and tasks. Java plugins that use the UDM Plugin-API from Deployit 3.7 will continue to work using a legacy layer. However, if your Java plugin extends classes from the Generic plugin or Python plugin, you may need to update your Java code, since these plugins now use the new API. Here's a table of replacements (read 'c.x.d.p.' as 'com.xebialabs.deployit.plugin')

Deployit 3.7	Deployit 3.8
c.x.d.p.api.deployment.execution.Step	c.x.d.p.api.flow.Step
c.x.d.p.api.deployment.execution.Step.Result	c.x.d.p.api.flow.StepExitCode
c.x.d.p.api.deployment.execution.DeploymentStep	c.x.d.p.api.flow.Step
c.x.d.p.api.deployment.execution.DeploymentExecutionContext	c.x.d.p.api.flow.ExecutionContext
c.x.d.p.api.inspection.InspectionExecutionContext	c.x.d.p.api.flow.ExecutionContext
c.x.d.p.api.inspection.InspectionPlanningContext	c.x.d.p.api.inspection.InspectionContext
c.x.d.p.overthere.ExecutionContextOverthereProcessOutputHandler	c.x.d.p.overthere.DefaultProcessOutputHandler

Upgrading to Deployit 3.7

- Deployit 3.7 contains an updated security implementation. Data from a pre-3.7 repository will automatically be converted to the new security model. See the section below for a complete description of the security upgrade.
- New versions of the following plugins are needed:
 - WAS
 - WLS
 - Tomcat
 - JBoss

Security upgrade

Permissions and repository

Starting with Deployit 3.7, Deployit needs to know the admin user's password to access the repository. This must be entered in the `deployit.conf` file prior to starting the new Deployit server. Note that the password in the configuration file is encrypted by the Deployit server when it starts.

The main change in the Deployit 3.7 security model is that local security permissions are stored only on root nodes and *core.Directory* CIs. During the upgrade, Deployit will convert pre-3.7-style permissions to the new permission system and automatically create directories to attach the relevant permissions to. The server log contains a description of the old security information that was found and how this was translated to the new permission system.

Permissions in Deployit are now assigned to **roles** instead of directly to principals. Deployit itself contains the definitions of roles and the principals they map to. Deployit will create a role for every user it finds during the upgrade.

It is also no longer possible to globally grant certain permissions. Permissions *deploy#initial*, *deploy#upgrade*, *deploy#undeploy*, *import#initial*, *import#upgrade*, *task#skip_step*, *task#move_step* can now only be granted on a directory CI. Deployit will convert these global permissions into directory permissions during the upgrade.

Deployit 3.7 no longer allows *deny* permissions. Instead, permissions should be segregated using the new *core.Directory* grouping CIs. The *deny* permissions are not migrated by Deployit and an equivalent configuration must be created manually.

Permission *repo#edit* no longer allows deleting of packages. From 3.7 onwards, new permission *import#remove* is required to be allowed to delete packages.

In summary, these are the security changes in Deployit 3.7:

Feature	Pre-3.7	3.7	Conversion
Admin password	Deployit did not need the administrator password.	Deployit needs the administrator password.	The administrator password must be set manually in the <i>deployit.conf</i> file before starting Deployit 3.7.
Local permissions defined per CI	Local permissions can be defined on any CI, if the permission is applicable.	Local permissions can only be defined on root nodes or core.Directory CIs.	Deployit automatically migrates permissions from pre-3.7 repositories to 3.7, creating new directory CIs as needed.
Permission assignment	Permissions are assigned to users (when using the repository as user store) or LDAP principals (users or groups).	Permissions are assigned to roles managed within Deployit.	Deployit automatically migrates principals from pre-3.7 repositories to 3.7, creating new roles as needed.
Permissions that are both global as well as local	Several permissions (<i>deploy#initial</i> , <i>deploy#upgrade</i> , <i>deploy#undeploy</i> , <i>import#initial</i> , <i>import#upgrade</i> , <i>task#skip_step</i> , <i>task#move_step</i>) can be assigned both globally as well as locally.	These permissions can only be assigned locally on root nodes or directory CIs.	Deployit automatically migrates these permissions from pre-3.7 repositories to 3.7.
deny permissions	Permissions can explicitly be denied on any level.	Denying permissions is no longer possible.	Deployit does not migrate deny permissions. An equivalent security configuration must be created manually after the upgrade.
Permission to delete packages	<i>repo#edit</i> permission allowed deleting of packages.	New permission <i>import#remove</i> is introduced to allow deleting of packages. Permission <i>repo#edit</i> is still required to edit CIs.	Deployit automatically assigns <i>import#remove</i> to roles that had <i>repo#edit</i> permission in the pre-3.7 repository.

LDAP

Configuration of LDAP security has also changed. The configuration in 3.7 is specified in a file called *deployit-security.xml*. The 3.6-style LDAP configuration in the *jackrabbit-jas.config* file can be migrated easily to the 3.7 format. The following table describes the mapping of the 3.6-style LDAP setup to the new 3.7 setup:

3.6 example	3.7 XML element	3.7 XML properties	3.7 example
<code>userProvider="ldap://localhost:389/ou=system"</code>	<code>security:ldap-server</code>	url and root properties	<code><security:ldap-server id="ldapServer" url="ldap://localhost:389/" root="ou=system"/></code>
<code>userFilter="(&(uid={USERNAME}))(objectClass=inetOrgPerson)"</code>	<code>security:ldap-authentication-provider</code>	user-search-filter	<code>user-search-filter="(&(uid={0}))(objectClass=inetOrgPerson)"</code>
<code>userSearchBase="dc=example,dc=com"</code>	<code>security:ldap-authentication-provider</code>	user-search-base	<code>user-search-base="dc=example,dc=com"</code>
<code>groupFilter="(memberUid={USERNAME})"</code>	<code>security:ldap-authentication-provider</code>	group-search-filter	<code>group-search-filter="(memberUid={0})"</code>
<code>groupSearchBase="ou=groups,dc=example,dc=com"</code>	<code>security:ldap-authentication-provider</code>	group-search-base	<code>group-search-base="ou=groups,dc=example,dc=com"</code>
<code>java.naming.security.principal="cn=admin,dc=example,dc=com"</code>	<code>security:ldap-server</code>	manager-dn	<code><security:ldap-server id="ldapServer" url="ldap://localhost:389/" manager-dn="cn=admin,dc=example,dc=com" manager-password="secret"/></code>
<code>java.naming.security.credentials="secret"</code>	<code>security:ldap-server</code>	manager-password	<code><security:ldap-server id="ldapServer" url="ldap://localhost:389/" manager-dn="cn=admin,dc=example,dc=com" manager-password="secret"/></code>
<code>useSSL</code>	<code>security:ldap-server</code>	url	Use the LDAPS protocol in the url, for example: <code><security:ldap-server id="ldapServer" url="ldaps://localhost:686"/></code>
<code>principalProvider=com.xebialabs.deployit.security.LdapPrincipalProvider</code>	n/a	n/a	This element is no longer required.

See the **System Administration manual** for more details on the available LDAP configuration options.

If you used Deployit 3.6 with LDAP, you will have modified the **conf/jackrabbit-repository.xml** file. For 3.7, the *Security* snippet must be:

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

  <LoginModule class="org.apache.jackrabbit.core.security.authentication.DefaultLoginModule">
    <param name="anonymousId" value="anonymous" />
    <param name="adminId" value="admin" />
  </LoginModule>
</Security>
```

Additional upgrade steps

After completing the upgrade procedure described in **Upgrading the Server** above, perform the following additional steps:

1. Edit the `deployit.conf` file and enter the admin password to property `admin.password`.
2. If you use LDAP, migrate the LDAP configuration to the `deployit-security.xml` configuration file.
3. Create a backup of your repository.
4. Start the Deployit server. When the server starts, Deployit will migrate the security information from the 3.6 repository to the 3.7 structure. In this process, Deployit will automatically create a role for each user in Deployit as well as special *core.Directory* CIs to assign permissions to.
5. Log into Deployit as the admin user and inspect the changes to the repository.
6. Open the Admin tab to create meaningful role assignments.
7. Open the Repository tab to create meaningful directory names.

Completing the migration

After the Deployit automated upgrade completes, log in to the Deployit GUI with various credentials to ensure that the security setup was correctly migrated. It is now possible to see the security permissions on *directory* CIs in the Repository Browser.

Upgrading to Deployit 3.6

- Deployit 3.6 is a drop-in replacement for Deployit 3.5. No repository migration is needed.
- New versions of the following plugins are needed:
 - WAS plugin
 - WLS plugin

Upgrading from Deployit 3.0 or earlier

Deployit and the plugins were changed extensively in version 3.5. As a consequence, it is **not** possible to automatically migrate from a 3.0 or earlier version to version 3.5 or later. If you want to perform this type of upgrade, please contact the XebiaLabs support team.