

# Deployit Generic Model Plugin Manual

Version 3.5.1

# Table of Content

Preface	3
Overview	3
Features	3
Requirements	3
Plugin Concepts	3
Container	3
Copied Artifact	3
Executed Script	3
Processed Template	4
Templating	4
Using the deployables and deployed	4
Deployable vs. Container Table	4
Deployed Actions Table	5
Sample Usage Scenario - Deploying a new middleware platform	5
Defining the Container	5
Defining a Configuration File	5
Defining a WAR	6
Defining a Datasource	6
CI Reference	6
Configuration Item Overview	6
Virtual Deployable Configuration Items	6
Virtual Deployed Configuration Items	7
Virtual Topology Configuration Items	7
Configuration Item Details	7
generic.AbstractDeployed	7
generic.AbstractDeployedArtifact	8
generic.Archive	9
generic.Container	10
generic.CopiedArtifact	10
generic.ExecutedScript	12
generic.File	13
generic.Folder	14
generic.ProcessedTemplate	14
generic.Resource	16

# Preface

This document describes the functionality provided by the Generic Model plugin.

See the **Deployit Reference Manual** for background information on Deployit and deployment concepts.

## Overview

Deployit supports a number of middleware platforms. Sometimes, though, it is necessary to extend Deployit with new middleware support. The Generic Model plugin provides a way to do this, without having to write Java code. Instead, using Deployit's flexible type system and the base CIs from the Generic Model plugin, new CIs can be defined by writing XML and providing scripts for functionality.

Several of Deployit's standard plugins are also built on top of the Generic Model plugin.

## Features

- Define custom containers
  - Stop, start, restart capabilities
- Define and copy custom artifacts to a custom container
- Define, copy and execute custom scripts on a custom container
- Define resources to be processed by a template and copied to a custom container
- Flexible templating engine

## Requirements

The plugin requires:

- **Deployit:** version 3.5+

## Plugin Concepts

The Generic Model plugin provides several CIs that can be used as base classes for creating Deployit extensions. There are base CIs for each of Deployit's CI types (deployables, deployeds and containers). A typical usage scenario is to create custom, synthetic CIs (based on one of the provided CIs) and using it to invoke the required behavior (scripts) in a deployment plan.

### Container

A [Container](#) is a topology CI and models middleware in your infrastructure. This would typically be used to model middleware for Deployit does not have out of the box support or that is custom to your environment. The other CIs in the plugin can be deployed to (subclasses of) the container. The behavior of the container in a deployment is configured by specifying scripts to be executed when it is started, stopped or restarted. Deployit will invoke these scripts as needed.

### Copied Artifact

A [CopiedArtifact](#) is an artifact as copied over to a [Container](#). It manages the copying of any generic artifact ([File](#), [Folder](#), [Archive](#), [Resource](#)) in the deployment package to the container. It is possible to indicate that this copied artifact requires a container restart.

### Executed Script

An [ExecutedScript](#) encapsulates a script that is executed on a generic container. The script is processed by the templating engine (see below) before being copied to the target container. The behavior of the script is configured by specifying scripts to be executed when it is deployed, upgraded or undeployed.

## Processed Template

A [ProcessedTemplate](#) is a [Freemarker](#) template that is processed by the templating engine (see below), then copied to a [Container](#).

## Templating

When defining and using CIs with the Generic Model plugin, the need arises to use variables in certain CI properties and scripts. The most obvious use is to include properties from the deployment itself, such as the names or locations of files in the deployment package. Deployit uses the [Freemarker](#) templating engine for this.

When performing a deployment using the Generic Model plugin, all CIs and scripts are processed in Freemarker. This means that placeholders can be used in CI properties and scripts to make them more flexible. Freemarker resolves placeholders using a *context*, a set of objects defining the template's environment. This context depends on the type of CI being deployed.

For deployed CIs, the context *deployed* refers to the current CI instance. For example:

```
<type type="tc.DeployedDataSource" extends="generic.ProcessedTemplate" deployable-type="tc.DataSource"
  container-type="tc.Server">
  ...
  <property name="targetFile" default="${deployed.name}-ds.xml" hidden="true"/>
  ...
</type>
```

For container CIs, the context *container* refers to the current container instance. For example:

```
<type type="tc.Server" extends="generic.Container">
  <property name="home" default="/tmp/tomcat"/>
  <property name="targetDirectory" default="${container.home}/webapps" hidden="true"/>
</type>
```

A special case is when referring to an artifact in the placeholder. For example, when deploying a CI representing a WAR file, the following placeholder can be used to refer to that file (assuming there is a *file* property on the deployable):

```
${deployed.deployable.file}
```

In this case, Deployit will copy the referred artifact to the target container so that the file is available to the executing script. A script containing a command like the following would therefore copy the file represented by the deployable to its installation path on the remote machine:

```
cp ${deployed.deployable.file} /install/path
```

## Using the deployables and deployed

### Deployable vs. Container Table

The following table describes which deployable / container combinations are possible. Note that the CIs can only be targeted to containers derived from [generic container](#).

Deployable	Containers	Generated deployed
generic.File generic.Archive	generic.Container	generic.CopiedArtifact
any deployable	generic.Container	generic.ExecutedScript
any deployable	generic.Container	generic.ProcessedTemplate

## Deployed Actions Table

The following table describes the effect a deployed has on its container.

Deployed	Create	Destroy	Modify
generic.CopiedArtifact	<ul style="list-style-type: none"> <li>Create target path on host, if needed</li> <li>Copy file to target path on host</li> </ul>	<ul style="list-style-type: none"> <li>Delete file from host</li> </ul>	<ul style="list-style-type: none"> <li>Delete old file from host</li> <li>Copy modified file to target path on host</li> </ul>
generic.ExecutedScript	<ul style="list-style-type: none"> <li>Run script through template engine</li> <li>Copy create script to container</li> <li>Execute script</li> </ul>	<ul style="list-style-type: none"> <li>Run script through template engine</li> <li>Copy destroy script to container</li> <li>Execute script</li> </ul>	<ul style="list-style-type: none"> <li>Run script through template engine</li> <li>Copy modify script to container</li> <li>Execute script</li> </ul>
generic.ProcessedTemplate	<ul style="list-style-type: none"> <li>Run script through template engine</li> <li>Copy template to container</li> </ul>	<ul style="list-style-type: none"> <li>Run script through template engine</li> <li>Delete template from container</li> </ul>	<ul style="list-style-type: none"> <li>Run script through template engine</li> <li>Delete template from container</li> <li>Copy new template to container</li> </ul>

## Sample Usage Scenario - Deploying a new middleware platform

This section describes an example of using the Generic Model plugin to implement support for a simple middleware platform. Deployment to this platform is done by simply copying a WAR archive to the right directory on the container. Resources are created by copying configuration files into the container's configuration directory. The Tomcat application server works in a very similar manner.

By defining a container and several other CIs based on CIs from the Generic Model plugin, it is possible to add support for deploying to this platform to Deployit.

### Defining the Container

To use any of the CIs in the Generic Model plugin, they need to be targeted to a [Container](#). This snippet shows how to define a generic container as a synthetic type:

```
<type type="tc.Server" extends="generic.Container">
  <property name="home" default="/tmp/tomcat"/>
</type>

<type type="tc.UnmanagedServer" extends="tc.Server">
  <property name="startScript" default="tc/start.sh" hidden="true"/>
  <property name="stopScript" default="tc/stop.sh" hidden="true"/>
  <property name="restartScript" default="tc/restart.sh" hidden="true"/>
</type>
```

Note that the *tc.UnmanagedServer* CI defines a start, stop and restart script. Deployit Server reads these scripts from the classpath. When targeting a deployment to the *tc.UnmanagedServer*, Deployit will include steps executing the start, stop and restart scripts in appropriate places in the deployment plan.

### Defining a Configuration File

The following snippet defines a CI based on the [CopiedArtifact](#). The *tc.DeployedFile* CI can be targeted to the *tc.Server*. The target directory is specified as a hidden property. Note the placeholder syntax used here.

```
<type type="tc.DeployedFile" extends="generic.CopiedArtifact" deployable-type="tc.File"
  container-type="tc.Server">
  <generate-deployable type="tc.File" extends="generic.File"/>
  <property name="targetDirectory" default="${deployed.container.home}/conf" hidden="true"/>
</type>
```

Using the above snippet, it is possible to create a package with a *tc.File* deployable and deploy it to an environment containing a *tc.UnmanagedServer*. This will result in a *tc.DeployedFile* deployed.

## Defining a WAR

To deploy a WAR file to the *tc.Server*, one possibility is to define a *tc.DeployedWar* CI that extends the [ExecutedScript](#). The *tc.DeployedWar* CI is generated when deploying a *jee.War* to the *tc.Server* CI. This is what the XML looks like:

```
<type type="tc.DeployedWar" extends="generic.ExecutedScript" deployable-type="jee.War"
  container-type="tc.Server">
  <generate-deployable type="tc.War" extends="jee.War"/>
  <property name="createScript" default="tc/install-war" hidden="true"/>
  <property name="modifyScript" default="tc/reinstall-war" hidden="true" required="false"/>
  <property name="destroyScript" default="tc/uninstall-war" hidden="true"/>
</type>
```

When performing an initial deployment, the create script, *tc/install-war* is executed on the target container. Inside the script, a reference to the *file* property is replaced by the actual archive. Note that the script files do not have an extension. Depending on the target platform, the extension *sh* (Unix) or *cmd* (Windows) is used.

## Defining a Datasource

Configuration files can be deployed by creating a CI based on the [ProcessedTemplate](#). By including a [Resource](#) in the package that is a Freemarker template, a configuration file can be generated during the deployment and copied to the container. This snippet defines such a CI, *tc.DeployedDataSource*:

```
<type type="tc.DeployedDataSource" extends="generic.ProcessedTemplate" deployable-type="tc.DataSource"
  container-type="tc.Server">
  <generate-deployable type="tc.DataSource" extends="generic.Resource"/>

  <property name="jdbcUrl"/>
  <property name="port" kind="integer"/>
  <property name="targetDirectory" default="${deployed.container.home}/webapps" hidden="true"/>
  <property name="targetFile" default="${deployed.name}-ds.xml" hidden="true"/>
  <property name="template" default="tc/datasource.ftl" hidden="true"/>
</type>
```

The *template* property specifies the Freemarker template file that Deployit Server reads from the classpath. The *targetDirectory* controls where the template is copied to. Inside the template, properties like *jdbcUrl* on the datasource can be used to produce a proper configuration file.

# CI Reference

## Configuration Item Overview

### Virtual Deployable Configuration Items

CI	Description
<a href="#">generic.Archive</a>	A generic, compressed binary artifact
<a href="#">generic.File</a>	A generic binary artifact
<a href="#">generic.Folder</a>	A generic folder artifact
<a href="#">generic.Resource</a>	A generic resource specification

## Virtual Deployed Configuration Items

CI	Description
<a href="#">generic.AbstractDeployed</a>	Abstract deployed that can target any deployable to a generic container
<a href="#">generic.AbstractDeployedArtifact</a>	Abstract deployed that can target any artifact to a generic container
<a href="#">generic.CopiedArtifact</a>	An artifact deployed on a generic container
<a href="#">generic.ExecutedScript</a>	A script executed on a generic container
<a href="#">generic.ProcessedTemplate</a>	A template deployed to a generic container

## Virtual Topology Configuration Items

CI	Description
<a href="#">generic.Container</a>	A container to which generic CIs can be deployed

## Configuration Item Details

### [generic.AbstractDeployed](#)

<b>Hierarchy</b>	udm.BaseDeployed >> udm.BaseConfigurationItem
<b>Interfaces</b>	udm.Deployed, udm.ConfigurationItem

Abstract deployed that can target any deployable to a generic container

Public Properties
<b>container</b> : CI<udm.Container>
The container on which this deployed runs.
<b>deployable</b> : CI<udm.Deployable>
The deployable that this deployed is derived from.

### Hidden Properties

**createOrder** : INTEGER = 50

The order of the step in the step list for the create operation.

**createVerb** : STRING = *Create*

Create Verb

**destroyOrder** : INTEGER = 40

The order of the step in the step list for the destroy operation.

**destroyVerb** : STRING = *Destroy*

Destroy Verb

**modifyOrder** : INTEGER = 50

The order of the step in the step list for the modify operation.

**modifyVerb** : STRING = *Modify*

Modify Verb

**noopOrder** : INTEGER = 50

The order of the step in the step list for the noop operation.

**noopVerb** : STRING = *Modify*

Noop Verb

**restartRequired** : BOOLEAN = *false*

The generic container requires a restart for the action performed by this deployed.

**restartRequiredForNoop** : BOOLEAN = *false*

The generic container requires a restart for the NOOP action performed by this deployed.

## generic.AbstractDeployedArtifact

**Hierarchy**    [generic.AbstractDeployed](#) >> [udm.BaseDeployed](#) >> [udm.BaseConfigurationItem](#)

**Interfaces**    [udm.Deployed](#), [udm.ConfigurationItem](#)

Abstract deployed that can target any artifact to a generic container

### Public Properties

**container** : CI<[udm.Container](#)>

The container on which this deployed runs.

**deployable** : CI<[udm.Deployable](#)>

The deployable that this deployed is derived from.

**targetFile** : STRING

Name of the artifact on the generic server.



## Hidden Properties

**createOrder** : INTEGER = 50

The order of the step in the step list for the create operation.

**createVerb** : STRING = *Create*

Create Verb

**destroyOrder** : INTEGER = 40

The order of the step in the step list for the destroy operation.

**destroyVerb** : STRING = *Destroy*

Destroy Verb

**modifyOrder** : INTEGER = 50

The order of the step in the step list for the modify operation.

**modifyVerb** : STRING = *Modify*

Modify Verb

**noopOrder** : INTEGER = 50

The order of the step in the step list for the noop operation.

**noopVerb** : STRING = *Modify*

Noop Verb

**targetDirectory** : STRING

Path to which artifact must be copied to on the generic server.

**createTargetDirectory** : BOOLEAN = *false*

Create the target directory on the generic server if it does not exist.

**restartRequired** : BOOLEAN = *false*

The generic container requires a restart for the action performed by this deployed.

**restartRequiredForNoop** : BOOLEAN = *false*

The generic container requires a restart for the NOOP action performed by this deployed.

**targetDirectoryShared** : BOOLEAN = *true*

Is the target directory shared by others on the generic server. When true, the target directory is not deleted during a destroy operation; only the artifacts copied to it.

## generic.Archive

**Hierarchy** udm.BaseDeployableArchiveArtifact >> udm.BaseDeployableFileArtifact >> udm.BaseDeployableArtifact >> udm.BaseDeployable >> udm.BaseConfigurationItem

**Interfaces** udm.Deployable, udm.SourceArtifact, udm.ArchiveArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FileArtifact

A generic, compressed binary artifact

## Public Properties

**placeholders** : SET\_OF\_STRING

Placeholders detected in this artifact

**Hidden Properties**

**textFileNamesRegex**\* : `STRING = .+\. (cfg | conf | config | ini | properties | props | txt | asp | aspx | htm | html | jsf | jsp | xht | xhtml | sql | xml | xsd | xsl | xslt)`

Regular expression that matches file names of text files

**generic.Container**

**Hierarchy** `udm.BaseContainer >> udm.BaseConfigurationItem`

**Interfaces** `udm.ConfigurationItem, udm.Container`

A container to which generic CIs can be deployed. Start, stop and restart behavior of this container can be controlled using the corresponding script properties.

**Public Properties**

**host**\* : `CI<overthere.Host>`

Host upon which the container resides

**Hidden Properties**

**restartOrder**\* : `INTEGER = 90`

The order of the restart container step in the step list.

**restartWaitTime**\* : `INTEGER = 0`

The time to wait in seconds for a container restart action.

**startOrder**\* : `INTEGER = 90`

The order of the start container step in the step list.

**startWaitTime**\* : `INTEGER = 0`

The time to wait in seconds for a container start action.

**stopOrder**\* : `INTEGER = 10`

The order of the stop container step in the step list.

**stopWaitTime**\* : `INTEGER = 0`

The time to wait in seconds for a container stop action.

**restartScript** : `STRING`

Classpath to the script used to restart the generic container.

**startScript** : `STRING`

Classpath to the script used to start the generic container.

**stopScript** : `STRING`

Classpath to the script used to stop the generic container.

**generic.CopiedArtifact**

**Hierarchy** `generic.AbstractDeployedArtifact >> generic.AbstractDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem`

**Interfaces** `udm.Artifact, udm.Deployed, udm.ConfigurationItem, udm.DerivedArtifact`

An artifact deployed on a generic container

### Public Properties

**container** : `CI<udm.Container>`

The container on which this deployed runs.

**deployable** : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

**placeholders** : `MAP_STRING_STRING`

A Map containing all the placeholders mapped to their values. Special values are <ignore> or <empty>

**targetFile** : `STRING`

Name of the artifact on the generic server.

## Hidden Properties

**createOrder** : INTEGER = 50

The order of the step in the step list for the create operation.

**createVerb** : STRING = *Create*

Create Verb

**destroyOrder** : INTEGER = 40

The order of the step in the step list for the destroy operation.

**destroyVerb** : STRING = *Destroy*

Destroy Verb

**modifyOrder** : INTEGER = 50

The order of the step in the step list for the modify operation.

**modifyVerb** : STRING = *Modify*

Modify Verb

**noopOrder** : INTEGER = 50

The order of the step in the step list for the noop operation.

**noopVerb** : STRING = *Modify*

Noop Verb

**targetDirectory** : STRING

Path to which artifact must be copied to on the generic server.

**createTargetDirectory** : BOOLEAN = *false*

Create the target directory on the generic server if it does not exist.

**restartRequired** : BOOLEAN = *false*

The generic container requires a restart for the action performed by this deployed.

**restartRequiredForNoop** : BOOLEAN = *false*

The generic container requires a restart for the NOOP action performed by this deployed.

**targetDirectoryShared** : BOOLEAN = *true*

Is the target directory shared by others on the generic server. When true, the target directory is not deleted during a destroy operation; only the artifacts copied to it.

## generic.ExecutedScript

**Hierarchy**    [generic.AbstractDeployed](#) >> [udm.BaseDeployed](#) >> [udm.BaseConfigurationItem](#)

**Interfaces**    [udm.Deployed](#), [udm.ConfigurationItem](#)

A script executed on a generic container

## Public Properties

**container** : `CI<udm.Container>`

The container on which this deployed runs.

**deployable** : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

## Hidden Properties

**createOrder** : `INTEGER = 50`

The order of the step in the step list for the create operation.

**createScript** : `STRING`

Classpath to the script that is uploaded and executed on the generic container for the create operation.

**createVerb** : `STRING = Create`

Create Verb

**destroyOrder** : `INTEGER = 40`

The order of the step in the step list for the destroy operation.

**destroyVerb** : `STRING = Destroy`

Destroy Verb

**modifyOrder** : `INTEGER = 50`

The order of the step in the step list for the modify operation.

**modifyVerb** : `STRING = Modify`

Modify Verb

**noopOrder** : `INTEGER = 50`

The order of the step in the step list for the noop operation.

**noopVerb** : `STRING = Modify`

Noop Verb

**destroyScript** : `STRING`

Classpath to the script that is uploaded and executed on the generic container for the destroy operation.

**modifyScript** : `STRING`

Classpath to the script that is uploaded and executed on the generic container for the modify operation.

**noopScript** : `STRING`

Classpath to the script that is uploaded and executed on the generic container for the noop operation.

**restartRequired** : `BOOLEAN = false`

The generic container requires a restart for the action performed by this deployed.

**restartRequiredForNoop** : `BOOLEAN = false`

The generic container requires a restart for the NOOP action performed by this deployed.

## generic.File

**Hierarchy** `udm.BaseDeployableFileArtifact >> udm.BaseDeployableArtifact >> udm.BaseDeployable >>`

udm.BaseConfigurationItem

**Interfaces** udm.Deployable, udm.SourceArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FileArtifact

A generic binary artifact

#### Public Properties

**placeholders** : SET\_OF\_STRING

Placeholders detected in this artifact

#### Hidden Properties

**textFileNamesRegex**\* : STRING = .+\.*(cfg | conf | config | ini | properties | props | txt | asp | aspx | htm | html | jsf | jsp | xht | xhtml | sql | xml | xsd | xsl | xslt)*

Regular expression that matches file names of text files

## generic.Folder

**Hierarchy** udm.BaseDeployableFolderArtifact >> udm.BaseDeployableArtifact >> udm.BaseDeployable >> udm.BaseConfigurationItem

**Interfaces** udm.Deployable, udm.SourceArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FolderArtifact

A generic folder artifact

#### Public Properties

**placeholders** : SET\_OF\_STRING

Placeholders detected in this artifact

#### Hidden Properties

**textFileNamesRegex**\* : STRING = .+\.*(cfg | conf | config | ini | properties | props | txt | asp | aspx | htm | html | jsf | jsp | xht | xhtml | sql | xml | xsd | xsl | xslt)*

Regular expression that matches file names of text files

## generic.ProcessedTemplate

**Hierarchy** generic.AbstractDeployedArtifact >> generic.AbstractDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem

**Interfaces** udm.Deployed, udm.ConfigurationItem

A template deployed to a generic container

## Public Properties

**container**\* : `CI<udm.Container>`

The container on which this deployed runs.

**deployable** : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

**targetFile** : `STRING`

Name of the artifact on the generic server.

## Hidden Properties

**createOrder**\* : `INTEGER = 50`

The order of the step in the step list for the create operation.

**createVerb**\* : `STRING = Create`

Create Verb

**destroyOrder**\* : `INTEGER = 40`

The order of the step in the step list for the destroy operation.

**destroyVerb**\* : `STRING = Destroy`

Destroy Verb

**modifyOrder**\* : `INTEGER = 50`

The order of the step in the step list for the modify operation.

**modifyVerb**\* : `STRING = Modify`

Modify Verb

**noopOrder**\* : `INTEGER = 50`

The order of the step in the step list for the noop operation.

**noopVerb**\* : `STRING = Modify`

Noop Verb

**targetDirectory**\* : `STRING`

Path to which artifact must be copied to on the generic server.

**template**\* : `STRING`

Classpath to the freemarker template used to generate the content of the final text base artifact.

**createTargetDirectory** : `BOOLEAN = false`

Create the target directory on the generic server if it does not exist.

**restartRequired** : `BOOLEAN = false`

The generic container requires a restart for the action performed by this deployed.

**restartRequiredForNoop** : `BOOLEAN = false`

The generic container requires a restart for the NOOP action performed by this deployed.

**targetDirectoryShared** : `BOOLEAN = true`

Is the target directory shared by others on the generic server. When true, the target directory is not deleted during a destroy operation; only the artifacts copied to it.

## generic.Resource

<b>Hierarchy</b>	udm.BaseDeployable >> udm.BaseConfigurationItem
<b>Interfaces</b>	udm.Deployable, udm.ConfigurationItem

A generic resource specification

---