

Deployit Generic Model Plugin Manual

Version 3.7.0-IF

Table of Content

Preface	3
Overview	3
Features	3
Requirements	3
Plugin Concepts	3
Container	3
NestedContainer	3
Copied Artifact	3
Executed Script	4
Executed Folder	4
Processed Template	4
Templating	4
Control Tasks	5
Using the deployables and deployed	5
Deployable vs. Container Table	5
Deployed Actions Table	5
Sample Usage Scenario - Deploying a new middleware platform	6
Defining the Container	6
Defining a Configuration File	6
Defining a WAR	7
Defining a Datasource	7
Discovery	7
Encoding	8
Property Inspection	8
Inspecting Set properties	8
Inspecting Map properties	9
Configuration Item Discovery	9
CI Reference	9
Configuration Item Overview	9
Topology Configuration Items	9
Virtual Deployable Configuration Items	10
Virtual Deployed Configuration Items	10
Virtual Topology Configuration Items	10
Configuration Item Details	10
generic.AbstractDeployed	10
generic.AbstractDeployedArtifact	11
generic.Archive	13
generic.Container	14
generic.CopiedArtifact	15
generic.ExecutedFolder	17
generic.ExecutedScript	19
generic.ExecutedScriptWithDerivedArtifact	22
generic.File	24
generic.Folder	24
generic.GenericContainer	25
generic.NestedContainer	25
generic.ProcessedTemplate	25
generic.Resource	28
overthere.CifsHost	28
overthere.Host	29
overthere.HostContainer	30
overthere.Jumpstation	30
overthere.LocalHost	32
overthere.SshHost	32

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 and folders on a custom container
- Define resources to be processed by a template and copied to a custom container
- Define and execute control tasks on containers and deployedes
- 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, deployedes 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.

Note: since Deployit version 3.6, the deployedes in the Generic Model Plugin can target containers that implement the [HostContainer](#) interface. In addition to the [Container](#) and derived CIs, this means they can also be targeted to CIs derived from [Host](#).

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.

NestedContainer

A [Nested Container](#) is a topology CI and models middleware in your infrastructure. The nested container allows for the modelling of abstract middleware concepts as containers to which items can be deployed.

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 [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.

Executed Folder

An [ExecutedFolder](#) encapsulates a folder containing installation and rollback scripts that are executed on a [Container](#). Installation scripts are executed when the folder is deployed or updated, rollback scripts are executed when it is undeployed. Execution of the scripts happens in order. Scripts are processed by the templating engine (see below) before being copied to the target container.

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
```

Control Tasks

CIIs based on the [Container](#), [CopiedArtifact](#), [ExecutedScript](#) or [ProcessedTemplate](#) can also define control tasks to manage them. Control tasks are implemented using scripts that will be loaded and executed on the target middleware when executed.

This is an example of defining a control task:

```
<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="pingScript" default="tc/ping.sh" hidden="true"/>
  <method name="ping" description="Test whether the datasource is available"/>
</type>
```

The *method* XML fragment defines that control task *ping* is available on the CI. The *pingScript* property indicates the script that must be executed when it is invoked. The script will be resolved from the classpath. The same mechanism applies to a container.

Using the deployables and deployments

Deployable vs. Container Table

The following table describes which deployable / container combinations are possible.

Deployable	Containers	Generated deployed
generic.File generic.Archive	overthere.HostContainer	generic.CopiedArtifact
any deployable	overthere.HostContainer	generic.ExecutedScript
any folder deployable	overthere.HostContainer	generic.ExecutedFolder
any deployable	overthere.HostContainer	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"> Create target path on host, if needed Copy file to target path on host 	<ul style="list-style-type: none"> Delete file from host 	<ul style="list-style-type: none"> Delete old file from host Copy modified file to target path on host
generic.ExecutedScript	<ul style="list-style-type: none"> Run script through template engine Copy create script to container Execute script 	<ul style="list-style-type: none"> Run script through template engine Copy destroy script to container Execute script 	<ul style="list-style-type: none"> Run script through template engine Copy modify script to container Execute script
generic.ExecutedFolder	<p>For each installation script in the folder (ordered alphabetically by name, ascending):</p> <ul style="list-style-type: none"> Run script through template engine Copy create script to container Execute script 	<p>For each rollback script in the folder (ordered alphabetically by name, descending):</p> <ul style="list-style-type: none"> Run script through template engine Copy destroy script to container Execute script 	<p>For each installation script in the folder that was not part of the deployment being upgraded (ordered alphabetically by name, ascending):</p> <ul style="list-style-type: none"> Run script through template engine Copy modify script to container Execute script
generic.ProcessedTemplate	<ul style="list-style-type: none"> Run script through template engine Copy template to container 	<ul style="list-style-type: none"> Run script through template engine Delete template from container 	<ul style="list-style-type: none"> Run script through template engine Delete template from container Copy new template to container

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 *bat* (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.

Discovery

The Generic Model plugin supports discovery in any subtype of [Container](#), [Nested Container](#) and [Deployed](#). Extenders of the plugin provide shell scripts that interact with the discovery mechanism, via the standard out, with specially formatted output representing the inspected property or discovered configuration item.

```

<!-- Sample of extending Generic Mode plugin -->
<type type="sample.TomcatServer" extends="generic.Container">
  ...
  <property name="inspectScript" default="inspect/inspect-server" hidden="true"/>
</type>

<type type="sample.VirtualHost" extends="sample.NestedContainer">
  <property name="server" kind="ci" as-containment="true" referenced-type="sample.TomcatServer"/>
  ...
  <property name="inspectScript" default="inspect/inspect-virtualhost" hidden="true"/>
</type>

<type type="sample.DataSource" extends="generic.ProcessedTemplate" deployable-type="sample.DataSourceSpec"
  container-type="sample.Server">
  <generate-deployable type="sample.DataSourceSpec" extends="generic.Resource"/>
  <property name="inspectScript" default="inspect/inspect-ds" hidden="true"/>
  ...
</type>

```

Encoding

The discovery mechanism uses URL encoding as described in [RFC3986](#) to interpret the value of an inspected property. It is the responsibility of the plugin extender to perform said encoding in the inspect shell scripts.

Sample of encoding in a BASH shell script

```

function encode()
{
  local myresult=$(printf "%b" "$1" | perl -pe 's/([^\s~A-Za-z0-9])/sprintf("%%02X", ord($1))/seg')
  echo "$myresult"
}
myString='This is a string spanning many lines and with funky characters like !@#%^&*() and \|"\'";:<>.[\{\}'
myEncodedString = $(encode "$myString")
echo $myEncodedString

```

Property Inspection

The discovery mechanism identifies an inspected property when output with the following format is sent to the standard out.

```
INSPECTED:propertyName=value
```

The output must be prefixed with *INSPECTED*: followed by the name of the inspected property, an = sign and then the encoded value of the property.

Sample :

```

echo INSPECTED:stringField=A,value,with,commas
echo INSPECTED:intField=1999
echo INSPECTED:boolField=true

```

Inspecting Set properties

When an inspected property is a set of strings, the value must be comma separated

```
INSPECTED:propertyName=value1,value2,value3
```


Sample :

```
echo INSPECTED:stringSetField=$(encode 'Jac,q,ues'),de,Molay
# will result in the following output
# INSPECTED:stringSetField=Jac%2Cq%2Cues,de,Molay
```

Inspecting Map properties

When an inspected property is a map of strings, entries must be comma separated and key values must be colon separated

```
INSPECTED:propertyName=key1:value1,key2:value2,key3:value3
```

Sample :

```
echo INSPECTED:mapField=first:$(encode 'Jac,q,ues:'),second:2
# will result in the following output
# INSPECTED:mapField=first:Jac%2Cq%2Cues,second:2
```

Configuration Item Discovery

The discovery mechanism identifies a discovered configuration item when output with the following format is sent to the standard out.

```
DISCOVERED:configurationItemId=type
```

The output must be prefixed with DISCOVERED:_ followed by the id of the configuration item as stored in the Deployit repository, an = sign and the type of the configuration item

Sample :

```
echo DISCOVERED:Infrastructure/tomcat/defaultContext=sample.VirtualHost
```

CI Reference

Configuration Item Overview

Topology Configuration Items

CI	Description
overthere.CifsHost	A machine that can be connected to using either WinRM or Telnet and can perform file manipulation via the CIFS protocol
overthere.Jumpstation	Description unavailable
overthere.LocalHost	The machine on which the Deployit Server is running on
overthere.SshHost	A machine that can be connected to using ssh

Virtual Deployable Configuration Items

CI	Description
generic.Archive	A generic, compressed binary artifact
generic.File	A generic binary artifact
generic.Folder	A generic folder artifact
generic.Resource	A generic resource specification

Virtual Deployed Configuration Items

CI	Description
generic.AbstractDeployed	Abstract deployed that can target any deployable to a generic container
generic.AbstractDeployedArtifact	Abstract deployed that can target any artifact to a generic container
generic.CopiedArtifact	An artifact deployed on a generic container
generic.ExecutedFolder	Scripts in the folder are executed against a Container based on a naming convention
generic.ExecutedScript	A script executed on a generic container
generic.ExecutedScriptWithDerivedArtifact	A script executed on a generic container whose deployable artifact supports placeholder replacement
generic.ProcessedTemplate	A template deployed to a generic container

Virtual Topology Configuration Items

CI	Description
generic.Container	A container to which generic CIs can be deployed
generic.GenericContainer	
generic.NestedContainer	A container that is nested with another container
overthere.Host	A machine that runs middleware, on which scripts can be executed, etc
overthere.HostContainer	


Configuration Item Details

[generic.AbstractDeployed](#)

Hierarchy udm.BaseDeployed >> udm.BaseConfigurationItem

Interfaces udm.Deployed, udm.ConfigurationItem

Abstract deployed that can target any deployable 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.	

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

inspectClasspathResources : SET_OF_STRING

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : STRING

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : SET_OF_STRING

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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.

inspectClasspathResources : SET_OF_STRING

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : STRING

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : SET_OF_STRING

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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.Taggable, udm.Deployable, udm.SourceArtifact, udm.ArchiveArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FileArtifact

A generic, compressed binary artifact

Public Properties

excludeFileNamesRegex : *STRING*

Regular expression that matches file names that must be excluded from scanning

placeholders : *SET_OF_STRING*

Placeholders detected in this artifact

scanPlaceholders : *BOOLEAN = true*

Scan Placeholders

tags : *SET_OF_STRING*

The tags to map deployables to containers.

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.Taggable, udm.ConfigurationItem, udm.Container, [generic.GenericContainer](#), [overthere.HostContainer](#)

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

envVars : *MAP_STRING_STRING*

Environment variables for container

tags : *SET_OF_STRING*

The tags to map deployables to containers.

Hidden Properties

restartOrder : INTEGER = 90

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

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.

inspectClasspathResources : SET_OF_STRING

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : STRING

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : SET_OF_STRING

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

restartScript : STRING

Classpath to the script used to restart the generic container.

restartWaitTime : INTEGER = 0

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

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.

inspectClasspathResources : SET_OF_STRING

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : STRING

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : SET_OF_STRING

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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.ExecutedFolder

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

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

Scripts in the folder are executed against a Container based on a naming convention

Public Properties



container : `CI<udm.Container>`

The container on which this deployed runs.

executorScript : `STRING`

Name of the executor script that will be executed for each script found in the folder.

rollbackScriptPostfix : `STRING`

A script's associated rollback script is derived by using the 1st group identified by the `scriptRecognitionRegex` and then appending this postfix to it. e.g give name '01-myscript.sql', regex '([0-9]*-*)\.sql' and rollback script postfix '-rollback.sql', we can derive the name of the associated rollback script to be '01-myscript-rollback.sql'

rollbackScriptRecognitionRegex : `STRING`

Regular expression used to identify a rollback script in the folder. A successful match should returns a single group, ie the logical script name. e.g. `[0-9]*-*.rollback\.sql`

scriptRecognitionRegex : `STRING`

Regular expression used to identify a script in the folder. A successful match should returns a single group to which the `rollbackScriptPostfix` can be appended to inorder to find the associated rollback script or the script's dependent subfolder. e.g. `([0-9]*-*)\.sql`

deployable : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

placeholders : `MAP_STRING_STRING`

A key/value pair mapping of placeholders in the deployed artifact to their values. Special values are and

Hidden Properties

commonScriptFolderName : *STRING = common*

Common folder that should be uploaded to the working directory.

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

classpathResources : *SET_OF_STRING*

Additional classpath resources that should be uploaded to the working directory before executing the script.

inspectClasspathResources : *SET_OF_STRING*

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : *STRING*

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : *SET_OF_STRING*

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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.

templateClasspathResources : *SET_OF_STRING*

Additional template classpath resources that should be uploaded to the working directory before executing the script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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

classpathResources : SET_OF_STRING

Additional classpath resources that should be uploaded to the working directory before executing the script.

destroyScript : STRING

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

inspectClasspathResources : SET_OF_STRING

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : STRING

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : SET_OF_STRING

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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.

remoteWorkingDirectoryPath : STRING

Name of working directory on target host. Default is to create a temporary directory which is deleted when connection is closed.

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.

retainRemoteWorkingDirectory : **BOOLEAN** = *false*

Retain the specified working directory on target host after completion.

templateClasspathResources : **SET_OF_STRING**

Additional template classpath resources that should be uploaded to the working directory before executing the script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.


generic.ExecutedScriptWithDerivedArtifact

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

Interfaces [udm.Artifact](#), [udm.Deployed](#), [udm.ConfigurationItem](#), [udm.DerivedArtifact](#)

A script executed on a generic container whose deployable artifact supports placeholder replacement

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 key/value pair mapping of placeholders in the deployed artifact to their values. Special values are and

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

classpathResources : SET_OF_STRING

Additional classpath resources that should be uploaded to the working directory before executing the script.

destroyScript : STRING

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

inspectClasspathResources : SET_OF_STRING

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : STRING

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : SET_OF_STRING

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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.

remoteWorkingDirectoryPath : STRING

Name of working directory on target host. Default is to create a temporary directory which is deleted when connection is closed.

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.

retainRemoteWorkingDirectory : **BOOLEAN** = *false*

Retain the specified working directory on target host after completion.

templateClasspathResources : **SET_OF_STRING**

Additional template classpath resources that should be uploaded to the working directory before executing the script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

generic.File

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

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

A generic binary artifact

Public Properties

excludeFileNamesRegex : **STRING**

Regular expression that matches file names that must be excluded from scanning

placeholders : **SET_OF_STRING**

Placeholders detected in this artifact

scanPlaceholders : **BOOLEAN** = *true*

Scan Placeholders

tags : **SET_OF_STRING**

The tags to map deployables to containers.

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.Taggable, udm.Deployable, udm.SourceArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FolderArtifact

A generic folder artifact

Public Properties**excludeFileNamesRegex** : **STRING**

Regular expression that matches file names that must be excluded from scanning

placeholders : **SET_OF_STRING**

Placeholders detected in this artifact

scanPlaceholders : **BOOLEAN** = *true*

Scan Placeholders

tags : **SET_OF_STRING**

The tags to map deployables to containers.

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.GenericContainer

null

generic.NestedContainer**Hierarchy** udm.BaseContainer >> udm.BaseConfigurationItem**Interfaces** udm.Taggable, udm.ConfigurationItem, udm.Container, [generic.GenericContainer](#), [overthere.HostContainer](#)

A container that is nested with another container

Public Properties**envVars** : **MAP_STRING_STRING**

Environment variables for container

tags : **SET_OF_STRING**

The tags to map deployables to containers.

Hidden Properties**inspectClasspathResources** : **SET_OF_STRING**

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : **STRING**

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : **SET_OF_STRING**

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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.

inspectClasspathResources : **SET_OF_STRING**

Additional classpath resources that should be uploaded to the working directory before executing the inspect script.

inspectScript : **STRING**

Classpath to the script used to inspect the generic container.

inspectTemplateClasspathResources : **SET_OF_STRING**

Additional template classpath resources that should be uploaded to the working directory before executing the inspect script. The template is first rendered and the rendered content copied to a file, with the same name as the template, in the working directory.

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

Hierarchy	udm.BaseDeployable >> udm.BaseConfigurationItem
Interfaces	udm.Tagable, udm.Deployable, udm.ConfigurationItem

A generic resource specification

Public Properties
tags : SET_OF_STRING The tags to map deployables to containers.

overthere.CifsHost

Hierarchy	overthere.Host >> udm.BaseContainer >> udm.BaseConfigurationItem
Interfaces	udm.Tagable, udm.ConfigurationItem, udm.Container, overthere.HostContainer

A machine that can be connected to using either WinRM or Telnet and can perform file manipulation via the CIFS protocol.

Public Properties
address : STRING Address of the host
connectionType : ENUM [TELNET, WINRM_HTTP, WINRM_HTTPS] = TELNET Connection Type
os : ENUM [WINDOWS, UNIX] Operating system
password : STRING Password to use for authentication
username : STRING Username to connect with
cifsPort : INTEGER = 445 Port on which the CIFS server runs
jumpstation : CI<overthere.Jumpstation> Jumpstation
port : INTEGER Port on which the Telnet or WinRM server runs
tags : SET_OF_STRING The tags to map deployables to containers.
temporaryDirectoryPath : STRING Directory into which temporary files are stored. Will be cleaned up when the connection is closed.

Hidden Properties

connectionTimeoutMillis : *INTEGER = 1200000*

Connection Timeout Millis

protocol : *STRING = cifs*

Protocol

tmpFileCreationRetries : *INTEGER = 1000*

Tmp File Creation Retries

winrmContext : *STRING = /wsman*

Winrm Context

winrmEnvelopSize : *INTEGER = 153600*

Winrm Envelop Size

winrmLocale : *STRING = en-US*

Winrm Locale

winrmTimeout : *STRING = PT60.000S*

Winrm Timeout

tmpDeleteOnDisconnect : *BOOLEAN = true*

Whether to delete the temporary connection directory when the connection is closed

Control Tasks

checkConnection

Check connection

overthere.Host

Hierarchy udm.BaseContainer >> udm.BaseConfigurationItem

Interfaces udm.Tagable, udm.ConfigurationItem, udm.Container, [overthere.HostContainer](#)

A machine that runs middleware, on which scripts can be executed, etc.

Public Properties

os : *ENUM [WINDOWS, UNIX]*

Operating system

jumpstation : *CI<overthere.Jumpstation>*

Jumpstation

tags : *SET_OF_STRING*

The tags to map deployables to containers.

temporaryDirectoryPath : *STRING*

Directory into which temporary files are stored. Will be cleaned up when the connection is closed.

Hidden Properties

connectionTimeoutMillis : **INTEGER** = *1200000*

Connection Timeout Millis

protocol : **STRING**

Protocol to use when connecting to this host

tmpFileCreationRetries : **INTEGER** = *1000*

Tmp File Creation Retries

tmpDeleteOnDisconnect : **BOOLEAN** = *true*

Whether to delete the temporary connection directory when the connection is closed

Control Tasks

checkConnection

Check connection

overthere.HostContainer

null

overthere.Jumpstation

Hierarchy [overthere.Host](#) >> udm.BaseContainer >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.ConfigurationItem, udm.Container, [overthere.HostContainer](#)

Description unavailable

Public Properties

address : *STRING*

Address of the host

os : *ENUM [WINDOWS, UNIX]*

Operating system

port : *INTEGER = 22*

Port on which the SSH server runs

username : *STRING*

Username to connect with

jumpstation : *CI<overthere.Jumpstation>*

Jumpstation

passphrase : *STRING*

Optional passphrase for the private key in the private key file

password : *STRING*

Password to use for authentication

privateKeyFile : *STRING*

Private key file to use for authentication

tags : *SET_OF_STRING*

The tags to map deployables to containers.

temporaryDirectoryPath : *STRING*

Directory into which temporary files are stored. Will be cleaned up when the connection is closed.

Hidden Properties

connectionTimeoutMillis : *INTEGER = 1200000*

Connection Timeout Millis

connectionType : *ENUM [SFTP, SFTP_CYGWIN, SFTP_WINSSHD, SCP, SUDO, INTERACTIVE_SUDO, TUNNEL] = TUNNEL*

Connection Type

interactiveKeyboardAuthRegex : *STRING = .*Password:[]?*

Regular expression to look for in keyboard-interactive authentication before sending the password

protocol : *STRING = ssh*

Protocol to use when connecting to this host

tmpFileCreationRetries : *INTEGER = 1000*

Tmp File Creation Retries

tmpDeleteOnDisconnect : *BOOLEAN = true*

Whether to delete the temporary connection directory when the connection is closed

Control Tasks**checkConnection**

Check connection

overthere.LocalHost**Hierarchy** [overthere.Host](#) >> udm.BaseContainer >> udm.BaseConfigurationItem**Interfaces** udm.Taggable, udm.ConfigurationItem, udm.Container, [overthere.HostContainer](#)

The machine on which the Deployit Server is running on.

Public Properties**os** : ENUM [WINDOWS, UNIX]

Operating system

jumpstation : CI<[overthere.Jumpstation](#)>

Jumpstation

tags : SET_OF_STRING

The tags to map deployables to containers.

temporaryDirectoryPath : STRING

Directory into which temporary files are stored. Will be cleaned up when the connection is closed.

Hidden Properties**connectionTimeoutMillis** : INTEGER = 1200000

Connection Timeout Millis

protocol : STRING = local

Protocol

tmpFileCreationRetries : INTEGER = 1000

Tmp File Creation Retries

tmpDeleteOnDisconnect : BOOLEAN = true

Whether to delete the temporary connection directory when the connection is closed

Control Tasks**checkConnection**

Check connection

overthere.SshHost**Hierarchy** [overthere.Host](#) >> udm.BaseContainer >> udm.BaseConfigurationItem**Interfaces** udm.Taggable, udm.ConfigurationItem, udm.Container, [overthere.HostContainer](#)

A machine that can be connected to using ssh.

Public Properties

address : *STRING*

Address of the host

connectionType : *ENUM [SFTP, SFTP_CYGWIN, SFTP_WINSSHD, SCP, SUDO, INTERACTIVE_SUDO, TUNNEL] = SFTP*

Type of SSH connection to create

os : *ENUM [WINDOWS, UNIX]*

Operating system

port : *INTEGER = 22*

Port on which the SSH server runs

username : *STRING*

Username to connect with

jumpstation : *CI<overthere.Jumpstation>*

Jumpstation

passphrase : *STRING*

Optional passphrase for the private key in the private key file

password : *STRING*

Password to use for authentication

privateKeyFile : *STRING*

Private key file to use for authentication

sudoUsername : *STRING*

Username to sudo to when accessing files or executing commands

tags : *SET_OF_STRING*

The tags to map deployables to containers.

temporaryDirectoryPath : *STRING*

Directory into which temporary files are stored. Will be cleaned up when the connection is closed.

Hidden Properties

connectionTimeoutMillis : *INTEGER* = *1200000*

Connection Timeout Millis

interactiveKeyboardAuthRegex : *STRING* = *.*Password:[]?*

Regular expression to look for in keyboard-interactive authentication before sending the password

protocol : *STRING* = *ssh*

Protocol

sudoCommandPrefix : *STRING* = *sudo -u {0}*

Sudo command to prefix to the original command. The placeholder {0} is replaced with the sudoUsername

sudoPasswordPromptRegex : *STRING* = *.*[Pp]assword.*:*

Regular expression to look for in interactive sudo before sending the password

tmpFileCreationRetries : *INTEGER* = *1000*

Tmp File Creation Retries

allocateDefaultPty : *BOOLEAN* = *false*

If true, a default pty is allocated when executing a command. All sudo implementations require it for interactive sudo, some even require it for normal sudo. Some SSH server implementations (notably the one on AIX 5.3) crash when it is allocated.

sudoOverrideUmask : *BOOLEAN* = *false*

If true, permissions are explicitly changed with `chmod -R go+rX` after uploading a file or directory with scp.

sudoQuoteCommand : *BOOLEAN* = *false*

If true, the original command is quoted when it is prefixed with sudoCommandPrefix

tmpDeleteOnDisconnect : *BOOLEAN* = *true*

Whether to delete the temporary connection directory when the connection is closed

Control Tasks

checkConnection

Check connection