

Deployit Database Plugin Manual

Version 3.6.1

Table of Content

Preface	3
Overview	3
Features	3
Requirements	3
Plugin Concepts	3
SQL Scripts	3
SQL Client	4
Usage in Deployment Packages	4
Using the deployables and deployed	4
Deployable vs. Container table	4
Deployed Actions Table	4
CI Reference	5
Configuration Item Overview	5
Deployable Configuration Items	5
Deployed Configuration Items	5
Topology Configuration Items	5
Virtual Topology Configuration Items	5
Configuration Item Details	5
sql.Db2Client	5
sql.ExecutedSqlScripts	7
sql.MySqlClient	9
sql.OracleClient	10
sql.SqlClient	12
sql.SqlScripts	13

Preface

This document describes the functionality provided by the database plugin.

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

Overview

The database plugin is a Deployit plugin that supports deployment of SQL files and folders to a database client.

Features

- Runs on Deployit 3.6 and up.
- Supports deployment to MySQL, Oracle and DB/2.
- Deploys and undeploys SQL files and folders.

Requirements

- **Deployit requirements**
 - **Deployit:** version 3.6+
 - **Other Deployit Plugins:** None
- **Infrastructural requirements**
 - **User credentials** for accessing the database client executables on the host running the database.

Plugin Concepts

SQL Scripts

The [SqlScripts](#) CI encompasses a folder containing SQL scripts that are to be executed on a database. SQL scripts come in two flavors, namely installation scripts and rollback scripts. Installation scripts are used to execute changes on the database, such as creation of a table or inserting data. Each installation script is associated with a rollback script which undoes the actions performed by it's companion installation script. Rollback scripts **must** have the exact same name as the installation script they are associated with and have the moniker `-rollback` attached to it. Executing an installation script followed by the accompanying rollback script should leave the database in an unchanged state.

SQL scripts are ordered alphabetically based on their filename. This is an example of ordering of several installation scripts:

- 1-create-user-table.sql
- 1-create-user-table-rollback.sql
- 10-drop-user-index.sql
- 10-drop-user-index-rollback.sql
- 2-insert-user.sql
- 2-insert-user-rollback.sql
- ...
- 9-create-user-index.sql
- 9-create-user-index-rollback.sql

Note that in this example, the tenth script, *10-drop-user-index.sql* would be incorrectly executed after the first script, *1-create-user-table.sql*.

When upgrading a SqlScripts CI, only those scripts that were not present in the previous package version are executed. For example, if the previous SqlScripts folder contained script1.sql and script2.sql, and the new version of SqlScripts folder contains script2.sql and script3.sql, then only script3.sql will be executed as part of the upgrade.

When undeploying a SqlScripts CI, all rollback scripts are executed in reverse alphabetical order.

SQL Client

The [SqlClient](#) CIs are containers to which [SqlScripts](#) can be deployed. The plugin ships with SqlClient for the following databases:

- MySQL
- Oracle
- DB/2

When SQL scripts are deployed to an SQL client, each script to be executed is run against the SQL client in turn. The SQL client can be configured with a username and password that is used to connect to the database. The credentials can be overridden on each SQL script if required.

Usage in Deployment Packages

The following is a manifest snippet that shows how SQL file and folder CIs can be included in a deployment package. The SQL scripts CI refers to a folder, *sql*, in the deployment package.

Manifest-Version: 1.0
Deployit-Package-Format-Version: 1.3
CI-Application: PetClinic-ear
CI-Version: 2.0
Name: PetClinic-2.0.ear
CI-Type: jee.Ear
CI-Name: PetClinic
Name: sql
CI-Type: sql.SqlScripts
CI-Name: sql

Using the deployables and deployed

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

Deployable vs. Container table

Deployable	Container	Generated deployed
sql.SqlScripts	sql.OracleClient, sql.MySqlClient, sql.Db2Client	sql.ExecutedSqlScripts

The following table describes the effect a deployed has on it's container.

Deployed Actions Table

Deployed	Actions performed for operations		
	Create	Destroy	Modify
sql.ExecutedSqlScripts	For each installation script in the folder (ordered alphabetically by name, ascending): <ul style="list-style-type: none"> • Run script through template 	For each rollback script in the folder (ordered alphabetically by name, descending): <ul style="list-style-type: none"> • Run script through template 	For each installation script in the folder that was not part of the deployment being upgraded (ordered alphabetically by name, ascending): <ul style="list-style-type: none"> • Run script through template engine

	engine <ul style="list-style-type: none"> • Copy create script to container • Execute script 	engine <ul style="list-style-type: none"> • Copy destroy script to container • Execute script 	<ul style="list-style-type: none"> • Copy modify script to container • Execute script
--	--	---	---

CI Reference

Configuration Item Overview

Deployable Configuration Items

CI	Description
sql.SqlScripts	Folder containing SQL scripts

Deployed Configuration Items

CI	Description
sql.ExecutedSqlScripts	SQL scripts executed on an SQL client

Topology Configuration Items

CI	Description
sql.Db2Client	IBM DB2 client
sql.MySqlClient	MySQL client
sql.OracleClient	Oracle SQL*Plus client

Virtual Topology Configuration Items

CI	Description
sql.SqlClient	Generic SQL client

Configuration Item Details

[sql.Db2Client](#)

Hierarchy [sql.SqlClient](#) >> generic.Container >> udm.BaseContainer >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.ConfigurationItem, generic.GenericContainer, udm.Container, overthere.HostContainer

IBM DB2 client

Public Properties

databaseName : *STRING*

The name of the DB2 database to connect to

db2Home : *STRING*

The directory that contains the DB2 installation



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

clientWrapperScript : *STRING* = *sql/Db2Client*

Client Wrapper Script

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.

password : *STRING*

Password

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.

username : *STRING*

Username

sql.ExecutedSqlScripts

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

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

SQL scripts executed on an SQL client

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.

password : STRING

If set, the password to use. Falls back to the optional default password set on the sql.SqlClient container

placeholders : MAP_STRING_STRING

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

username : STRING

If set, the user name to use. Falls back to the optional default user name set on the sql.SqlClient container

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 = Run*

Create Verb

destroyOrder : *INTEGER = 40*

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

destroyVerb : *STRING = Rollback*

Destroy Verb

executorScript : *STRING = \${deployed.container.clientWrapperScript}*

Executor Script

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

rollbackScriptPostfix : *STRING = -rollback.sql*

Rollback Script Postfix

rollbackScriptRecognitionRegex : *STRING = [0-9]*-(.*)-rollback\.sql*

Rollback Script Recognition Regex

scriptRecognitionRegex : *STRING = (?!.*-rollback\.sql)([0-9]*-.*)\.sql*

Script Recognition Regex

classpathResources : *SET_OF_STRING*

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

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.

sql.MySqlClient

Hierarchy [sql.SqlClient](#) >> generic.Container >> udm.BaseContainer >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.ConfigurationItem, generic.GenericContainer, udm.Container, overthere.HostContainer

MySQL client

Public Properties

databaseName : [STRING](#)

The name of the MySQL database to connect to



host : [CI<overthere.Host>](#)

Host upon which the container resides

mySqlHome : [STRING](#)

The directory that contains the MySQL installation

envVars : [MAP_STRING_STRING](#)

Environment variables for container

password : [STRING](#)

If set, the password to use if none is set on the deployed sql.ExecutedSqlScripts

tags : [SET_OF_STRING](#)

The tags to map deployables to containers.

username : [STRING](#)

If set, the user name to use if none is set on the deployed sql.ExecutedSqlScripts

Hidden Properties

clientWrapperScript : *STRING* = *sql/MySqlClient*

Client Wrapper Script

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.

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.

sql.OracleClient

Hierarchy [sql.SqlClient](#) >> generic.Container >> udm.BaseContainer >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.ConfigurationItem, generic.GenericContainer, udm.Container, overthere.HostContainer

Oracle SQL*Plus client

Public Properties



host : `CI<overthere.Host>`

Host upon which the container resides

oraHome : `STRING`

The directory that contains the Oracle installation

sid : `STRING`

The Oracle SID to connect to

envVars : `MAP_STRING_STRING`

Environment variables for container

password : `STRING`

If set, the password to use if none is set on the deployed `sql.ExecutedSqlScripts`

tags : `SET_OF_STRING`

The tags to map deployables to containers.

username : `STRING`

If set, the user name to use if none is set on the deployed `sql.ExecutedSqlScripts`

Hidden Properties

clientWrapperScript : `STRING` = *sql/OracleClient*

Client Wrapper Script

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.

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.

sql.SqlClient

Hierarchy generic.Container >> udm.BaseContainer >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.ConfigurationItem, generic.GenericContainer, udm.Container, overthere.HostContainer

Generic SQL client

Public Properties

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

Host upon which the container resides

clientWrapperScript : STRING

The OS-specific wrapper script that calls the SQL client

envVars : MAP_STRING_STRING

Environment variables for container

password : STRING

If set, the password to use if none is set on the deployed sql.ExecutedSqlScripts

tags : SET_OF_STRING

The tags to map deployables to containers.

username : STRING

If set, the user name to use if none is set on the deployed sql.ExecutedSqlScripts

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.

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.

sql.SqlScripts

Hierarchy generic.Folder >> udm.BaseDeployableFolderArtifact >> udm.BaseDeployableArtifact >> udm.BaseDeployable >> udm.BaseConfigurationItem

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

Folder containing SQL scripts

Public Properties

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