

Deployit Database Plugin Manual

Version 3.8.3

Table of Contents

Table of Contents	2
Preface	3
Overview	3
Features	3
Requirements	3
Plugin Concepts	3
SQL Scripts	3
Dependencies	4
SQL Client	4
Usage in Deployment Packages	4
Using the deployables and deployed	5
Deployable vs. Container table	5
Deployed Actions Table	5
CI Reference	5
Configuration Item Overview	5
Deployables	5
Deployeds	5
Containers	5
Other Configuration Items	5
Configuration Item Details	5
sql.Db2Client	6
sql.ExecutedSqlScripts	6
sql.MsSqlClient	8
sql.MySqlClient	9
sql.OracleClient	10
sql.SqlClient	11
sql.SqlScripts	12

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.

Dependencies

It is also possible to include dependencies together with the SQL scripts. Dependencies are included in the package using sub-folders. Sub-folders that have the same name as the script (without the file extension) are uploaded with the scripts in the sub-folder along with the main script to the target machine. The main script can then execute the dependent scripts in the same connection.

Common dependencies can be included in a sub-folder called `common` and will be available to all scripts.

Take this folder with Oracle scripts as an example:

```
mysqlfolder
|
|__ 01_CreateTable.sql
|
|__ 02_CreateUser.sql
|
|__ 02_CreateUser
|
|__ create_admin_users.sql
|
|__ create_power_users.sql
|
|__ common
|
|__ some_other_util.sql
|
|__ some_resource.properties
```

The `02_CreateUser.sql` script can use its dependencies or common dependencies as follows:

```
--
-- 02_CreateUser.sql
--

INSERT INTO person2 (id, firstname, lastname) VALUES (1, 'xebialabs1', 'user1');
-- Execute a common dependency
@some_other_util.sql
-- Execute script-specific dependency: Create Admin Users
@create_admin_users.sql
-- Execute script-specific dependency: Create Power Users
@create_power_users.sql
COMMIT;
```

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 engine Copy create script to container Execute script 	For each rollback script in the folder (ordered alphabetically by name, descending): <ul style="list-style-type: none"> Run script through template engine Copy destroy script to container Execute script 	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 Copy modify script to container Execute script

CI Reference

Configuration Item Overview

Deployables

CI	Description
sql.SqlScripts	Folder containing SQL scripts

Deployeds

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

Containers

CI	Description
sql.Db2Client	IBM DB2 client
sql.MsSqlClient	Microsoft SQL Server client
sql.MySqlClient	MySQL client
sql.OracleClient	Oracle SQL*Plus client
sql.SqlClient	Generic SQL client

Other Configuration Items

CI	Description
sql.Db2Client	IBM DB2 client
sql.ExecutedSqlScripts	SQL scripts executed on an SQL client
sql.MsSqlClient	Microsoft SQL Server client
sql.MySqlClient	MySQL client
sql.OracleClient	Oracle SQL*Plus client
sql.SqlClient	Generic SQL client
sql.SqlScripts	Folder containing SQL scripts

Configuration Item Details

sql.Db2Client

Type Hierarchy `sql.SqlClient` >> `generic.Container` >> `udm.BaseContainer` >> `udm.BaseConfigurationItem`

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

IBM DB2 client

Parent	
* host : <code>CI<overthere.Host></code>	Host upon which the container resides
Public Properties	
* databaseName : <code>STRING</code>	The name of the DB2 database to connect to
* db2Home : <code>STRING</code>	The directory that contains the DB2 installation
envVars : <code>MAP_STRING_STRING</code>	Environment variables for container
tags : <code>SET_OF_STRING</code>	If set, only deployables with the same tag will be automatically mapped to this container.
Hidden Properties	
* clientWrapperScript : <code>STRING</code> = <code>sql/Db2Client</code>	Client Wrapper Script
* restartOrder : <code>INTEGER</code> = 90	The order of the restart container step in the step list.
* startOrder : <code>INTEGER</code> = 90	The order of the start container step in the step list.
* startWaitTime : <code>INTEGER</code> = 0	The time to wait in seconds for a container start action.
* stopOrder : <code>INTEGER</code> = 10	The order of the stop container step in the step list.
* stopWaitTime : <code>INTEGER</code> = 0	The time to wait in seconds for a container stop action.
inspectClasspathResources : <code>SET_OF_STRING</code>	Additional classpath resources that should be uploaded to the working directory before executing the inspect script.
inspectScript : <code>STRING</code>	Classpath to the script used to inspect the generic container.
inspectTemplateClasspathResources : <code>SET_OF_STRING</code>	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.
password : <code>STRING</code>	Password
restartScript : <code>STRING</code>	Classpath to the script used to restart the generic container.
restartWaitTime : <code>INTEGER</code> = 0	The time to wait in seconds for a container restart action.
startScript : <code>STRING</code>	Classpath to the script used to start the generic container.
stopScript : <code>STRING</code>	Classpath to the script used to stop the generic container.
username : <code>STRING</code>	Username

sql.ExecutedSqlScripts

Type 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

Parent

* **container** : `CI<udm.Container>`
The container on which this deployed runs.

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.
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.

sql.MsSqlClient

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

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

Microsoft SQL Server client

Parent	
* host : CI < overthere.Host >	Host upon which the container resides

Public Properties	
* serverName : STRING	The name of the MS SQL Server to connect to
databaseName : STRING	The name of the MS SQL database to connect to
envVars : MAP_STRING_STRING	Environment variables for container
password : STRING	Password
tags : SET_OF_STRING	If set, only deployables with the same tag will be automatically mapped to this container.
username : STRING	Username
Hidden Properties	
* clientWrapperScript : STRING = sql/MsSqlClient	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.
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.

sql.MySqlClient

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

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

MySQL client

Parent	
* host : Cl < overthere.Host >	Host upon which the container resides

Public Properties	
* databaseName : STRING	The name of the MySQL database to connect to
* 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	If set, only deployables with the same tag will be automatically mapped to this container.
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.
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.

sql.OracleClient

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

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

Oracle SQL*Plus client

Parent	
* host : CI < overthere.Host >	Host upon which the container resides

Public Properties	
* 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	If set, only deployables with the same tag will be automatically mapped to this container.
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.
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.

sql.SqlClient

Virtual Type

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

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

Generic SQL client

Parent	
* host : Cl<overthere.Host>	Host upon which the container resides

Public Properties	
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	If set, only deployables with the same tag will be automatically mapped to this container.
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.
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.

sql.SqlScripts

Type 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	
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	Whether to scan this artifact for placeholders when it is imported
tags : SET_OF_STRING	If set, this deployable will only be mapped automatically to containers with the same tag.

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