Installing a JDBC Driver

DbVisualizer bundles JDBC drivers for most common databases, so typically you do not need to install a JDBC driver.

This page describes the way JDBC drivers are managed in DbVisualizer. If a JDBC driver for your database is bundled with DbVisualizer, see Driver Info on the Supported Databases page, you typically do not need to read this chapter.

If, however, any of the these things apply to you, keep on reading:

- want to learn how the Driver Manager in DbVisualizer works,
- need to have several versions of the same JDBC driver loaded simultaneously,
- need to add a Driver that does not exist in the list of default drivers .

What is a JDBC Driver?

DbVisualizer is a generic tool for administration and exploration of databases. DbVisualizer does not deal directly with how to communicate with each database type. That job is done by a JDBC driver, which is a set of Java classes. All JDBC drivers conform to the JDBC specification and its standardized Java programming interfaces. This is what DbVisualizer relies on. A JDBC driver implements all details for how to communicate with a specific database and database version, and there are drivers available from the database vendors themselves as well as from third parties. To establish a connection to a database, DbVisualizer loads the driver and then gets connected to the database through the driver.

The following sections describe the steps for installing a JDBC Driver, and also how to configure DbVisualizer to use JNDI to obtain a database connection.

Get the JDBC driver file(s)

DbVisualizer comes bundled with ready-to-use drivers in predefined templates for the most commonly used JDBC drivers that have licenses that allow for distribution with a third party product. Currently, drivers for Azure SQL Database, Db2, Greenplum, H2, JavaDB/Derby, Mimer SQL, MySQL, NuoDB, Oracle, PostgreSQL, SQLite, Vertica, Yellowbrick as well the jTDS driver for SQL Server and Sybase, are included with DbVisualizer. If you only need to connect to databases of these types, you can skip the rest of this chapter and jump straight to the Creating a Connection page, because by default, DbVisualizer creates a user configuration for these drivers automatically when you create a connection.

The Driver Manager is used to create driver configurations. If you need to connect to a database that is not supported by a bundled JDBC driver template, you must create a user configuration for a JDBC driver that works with your database type and version. This is done from a predefined template in the Driver Manager. The following web page contains an up-to-date listing of the database/driver combinations we have tested and supports via templates:

http://www.dbvis.com/doc/database-drivers/

For other databases/drivers you need to use the Custom Template and do all the configuration. To find a JDBC driver for your database, go to the database vendor's website or search for the name of the database plus the word `JDBC`. Many drivers are accessible from Maven or HTTP download sites. A good place too search for drivers distributed with Maven is https://mvnrepository.com/

You can download Drivers that is accessible via Maven or HTTP directly in the Driver Manger.

For proprietary drivers you need to download the driver to an appropriate directory. Make sure to read the installation instructions provided with the driver. Some drivers are delivered in ZIP or JAR format. ZIP files need to be unpacked to make the driver files loadable in the Driver Manager. The Databases and JDBC Drivers web page describes where you can download some drivers and also what additional steps may be needed to install and load the driver in DbVisualizer.

Drivers are categorized into 4 types. We're not going to explain the differences here, just give you the hint that the "type 4," aka "thin," drivers are the easiest to maintain, since they are pure Java drivers and do not depend on any external DLL's or dynamic libraries. Even though DbVisualizer works with any type of driver, we recommend that you get a type 4 driver if there is one for your database.

When you have downloaded and configured the JDBC driver in Driver Manager, you can go ahead and create a database connection, as described in the Creating a Connection page.

Driver Manager

The **Driver Manager** in DbVisualizer is used to define and configure the drivers that will be used to communicate with the databases. DbVisualizer comes with predefined configuration templates that is used to create the user driver configurations that will be used in connections. Templates may be updated in new versions of DbVisualizer, but your user configuration will not be changed. The templates may contain:

- Ready-to-use configuration with bundled driver files
- Predefined configuration with downloadable artifacts (you will have to download driver files via the artifacts from in the Driver Manager)

- Just a configuration (you will have to add driver files or artifacts)

The Custom Template has no configuration and can be used to create a driver for a database the we have not tested yet

Loading and Configuring Drivers Manually

You can also load and configure JDBC drivers manually using the Driver Manager.

The left part of the driver manager dialog contains a list of driver names with a status symbol indicating whether the driver has been configured (green checked icon) or not. The list columns also shows how many connections that uses the driver and the version of the driver. The list shows both the current user drivers and the templates (read only). You can search and sort the list. Initially, the driver list contains a collection of templates. The list is used to maintain the drivers and you may:

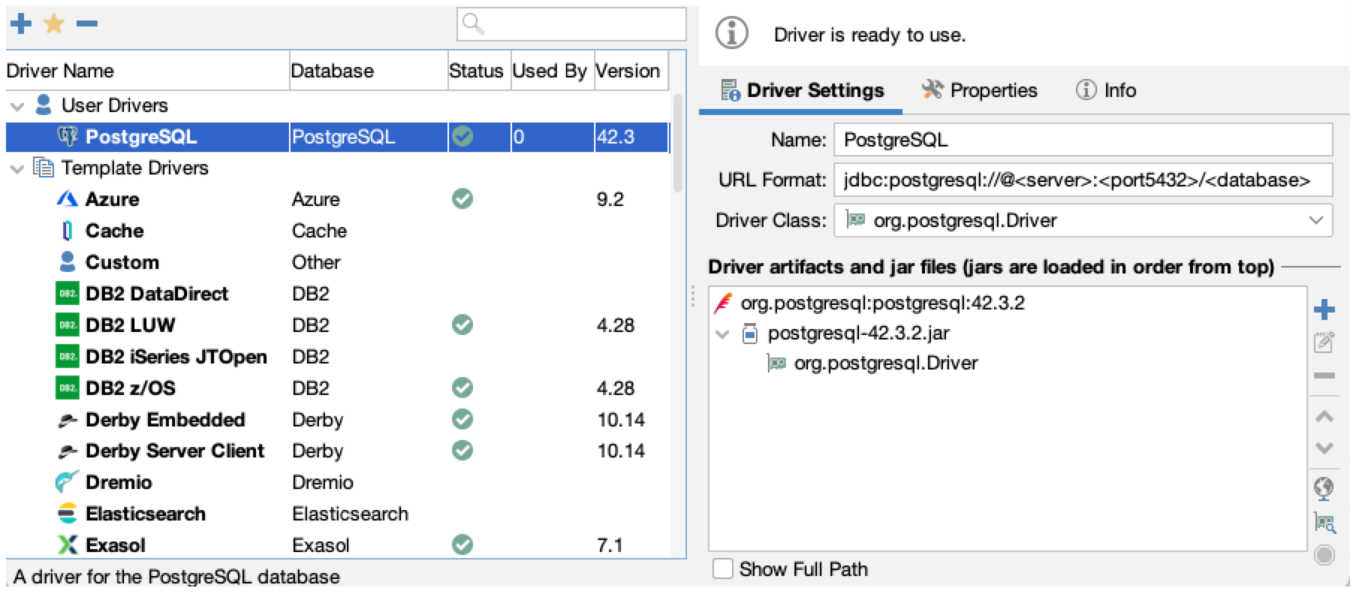
- Create a new user driver from a template (plus)
- Copy an existing user driver (star)
- Delete an existing user driver (minus)

The right part displays the driver configuration for the selected driver in terms of the following:

- **Information area**
  Presents information about the current state if the driver and what to do next if the driver is not ready-to-use.
- **Separate tabs for**
  - **Driver Settings** - described in detail below
  - **Properties** - setting for driver properties that is common for all connections that uses the driver configuration (settings can still be changed on the individual connections)
  - **Info**rmation - shows general information about the driver

The most important tab is the **Driver Settings** where you have:

- **Name**
  A driver name in the scope of DbVisualizer is a logical name for either a JDBC driver that is shown when you create a connection and in the Connection tab when selecting which driver to use for a Database Connection
- **URL Format**
  The URL format specifies the pattern for the JDBC URL. Its purpose is to assist the user in the Connection tab when entering URL information. See Using Variables in Connection Fields for more about how you can make it really easy to create Database Connections for this driver later on. For a custom driver the JDBC URL is used configure the connection.
- **Driver Class**
  Defines the main class for the JDBC driver, used for connecting to the database.
- **Driver artifacts and jar files (files tree)**
  Defines artifact to download driver files from Maven or HTTP or paths to search for JDBC drivers.



Driver Name	Database	Status	Used By	Version
User Drivers				
PostgreSQL	PostgreSQL	✓	0	42.3
Template Drivers				
Azure	Azure	✓		9.2
Cache	Cache			
Custom	Other			
DB2 DataDirect	DB2			
DB2 LUW	DB2	✓		4.28
DB2 iSeries JTOpen	DB2			
DB2 z/OS	DB2	✓		4.28
Derby Embedded	Derby	✓		10.14
Derby Server Client	Derby	✓		10.14
Dremio	Dremio			
Elasticsearch	Elasticsearch			
Exasol	Exasol	✓		7.1

A driver for the PostgreSQL database

Driver is ready to use.

Driver Settings | Properties | Info

Name: PostgreSQL

URL Format: jdbc:postgresql://@<server>:<port5432>/<database>

Driver Class: org.postgresql.Driver

Driver artifacts and jar files (jars are loaded in order from top)

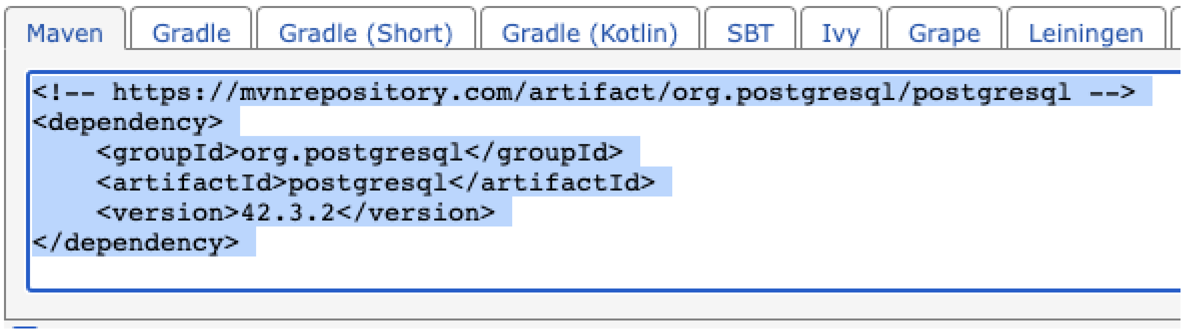
```
org.postgresql:postgresql:42.3.2
postgresql-42.3.2.jar
    org.postgresql.Driver
```

Show Full Path

A driver is ready to use once a driver class has been identified, which is indicated with a green check icon in the list. Drivers that are not ready for use are shown without an icon, or with a red cross icon if an error has been detected (such as a failed download or a missing file) .

Setup a JDBC driver

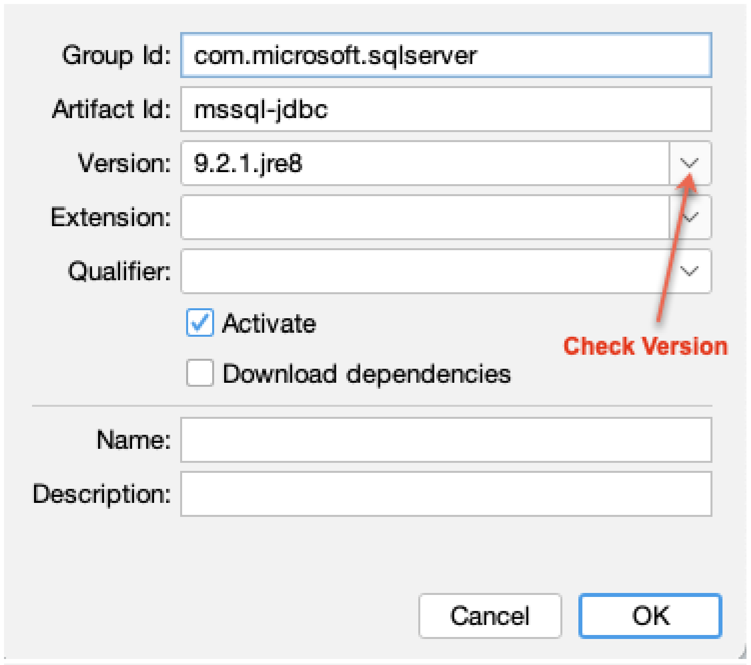
The recommended way to setup a predefined driver without bundled driver files is to pick a matching driver template from the list and create a user driver and then add artifacts, files or a or directory that keeps the driver class(es). For instance, if you are going to load the JDBC driver for **Db2 DataDirect**, selec t the corresponding driver entry in the list and press the plus icon (or use the context menu). You can also copy an existing user driver. Artifacts (HTTP, or Maven), files and folders can be added via drag&drop, copy&paste or via the plus icon on the right side that opens editors or a file dialog). Maven artifacts are identified by the dependency that can be found in e.g. https://mvnrepository.com/ (copy&paste/drag&drop this to the left area or to the Maven editor)



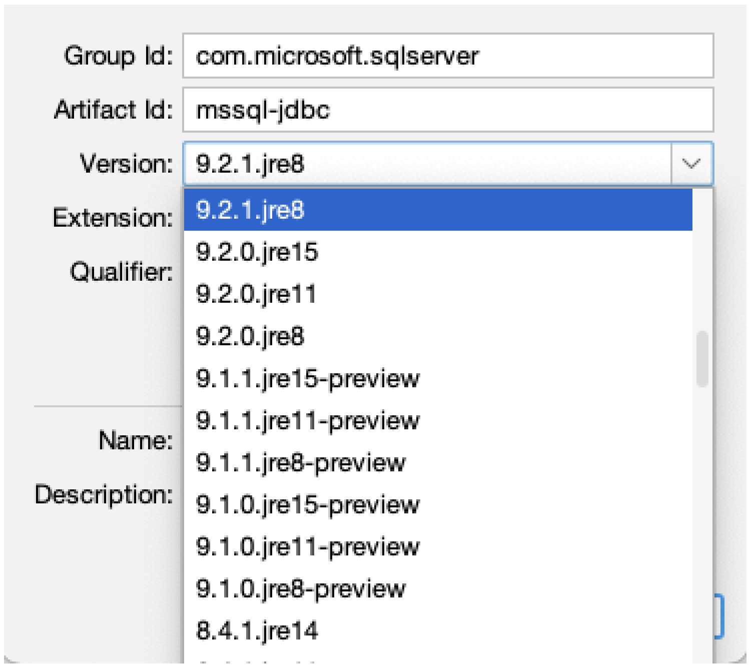
```
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.3.2</version>
</dependency>
```

Artifacts must be downloaded via the globe icon on the right. The dowloaded or added files/folders, are scanned and if all works well a a green checkbox is displayed at the driver in the list

The preferred way to download is via Maven. The Maven editor allows you to check what versions exist. Both the Maven and the HTTP editors allows to set a name and a description.

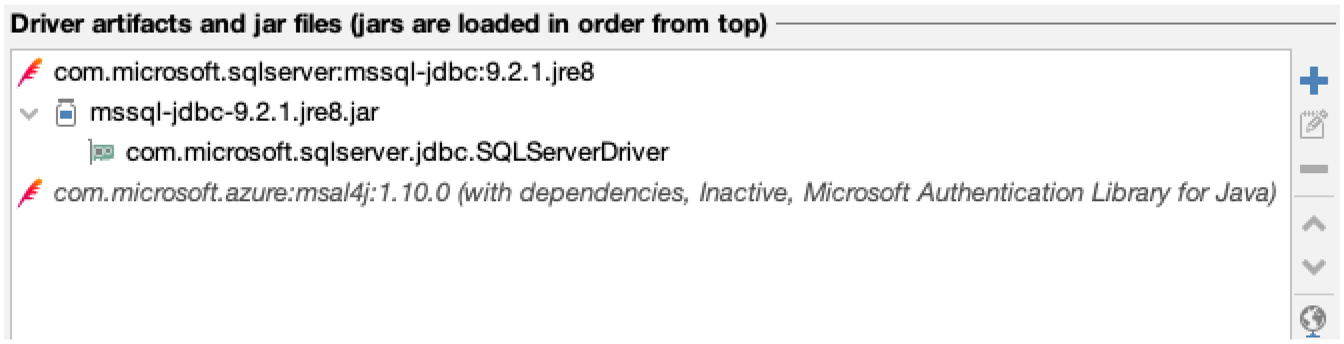


Group Id: com.microsoft.sqlserver
Artifact Id: mssql-jdbc
Version: 9.2.1.jre8
Extension:
Qualifier:
[✓] Activate
[ ] Download dependencies
Name:
Description:



Group Id: com.microsoft.sqlserver
Artifact Id: mssql-jdbc
Version: 9.2.1.jre8
Extension:
Qualifier:
Name:
Description:

Artifacts can be set inactive i.e. not used in download. This is mainly used in templates for artifacts that should only be used for specific connection purposes e.g. special authentication. Inactive artifacts are displayed in italic and in a disabled color:


Driver artifacts and jar files (jars are loaded in order from top)

com.microsoft.sqlserver:mssql-jdbc:9.2.1.jre8
mssql-jdbc-9.2.1.jre8.jar
com.microsoft.sqlserver.jdbc.SQLServerDriver
com.microsoft.azure:msal4j:1.10.0 (with dependencies, Inactive, Microsoft Authentication Library for Java)

Check the following online web page with the most current information about the tested databases and drivers:

http://www.dbvis.com/doc/database-drivers/

- It lists which databases and drivers we have tested
- Download links to JDBC drivers
- Information about which files to load in the driver manager for each JDBC driver
- Information about which Driver Class to choose

A JDBC Driver implementation typically consists of several Java classes. Java classes are typically organized using a package name structure. Example:

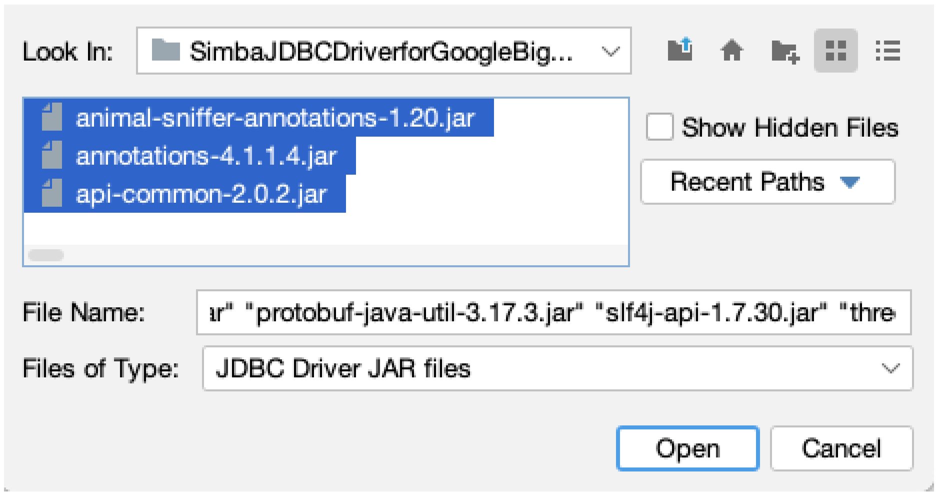
```
oracle.jdbc.driver.OracleDriver
```

When you load files in the Drivers artifacts and files tree, DbVisualizer **scans** each file to find the classes that represent main driver classes. Each such class is listed under the path where it was found in the Driver JAR Files list, and it is also added to the **Driver Class** list in the Driver Settings area above. If there is more than one class in the list, make sure you select the correct Driver Class from the list. Consult the driver documentation (or the Databases and JDBC Drivers page) for information about which class to select.

When Driver Manager scans the loaded files in Drivers artifacts and files tree, the files are searched from the top of the tree, i.e., if there are several identical classes, the topmost class will be used. Loading several paths containing different versions of the same driver in one driver definition is not recommended, even though it works (if you do this, you must move the driver you are going to use to the top of the tree). The preferred method for handling multiple versions of a driver is to create several user driver definitions.

JDBC drivers that require several JAR files

Some drivers depend on several JAR files, or directories. Simply select all JARs at once and use copy&paste, drag&drop or the file dialog to add them. The Driver Manager will then automatically scan each of the loaded files and present any JDBC driver classes it finds.



animal-sniffer-annotations-1.20.jar
annotations-4.1.1.4.jar
api-common-2.0.2.jar

File Name: ır" "protobuf-java-util-3.17.3.jar" "slf4j-api-1.7.30.jar" "thre

Files of Type: JDBC Driver JAR files

Errors (why are some paths red?)

A path in red color indicates that the path is invalid. This may happen if the path has been removed or moved after it was loaded into the driver manager. Simply remove the erroneous path and locate the correct one.

Several versions of the same driver

The Driver Manager supports loading and using several versions of the same driver concurrently. We recommend that you create a unique user driver configuration per version of the driver and name the driver configuration properly, e.g., **Oracle 9.2.0.1**, **Oracle 10.2.1.0.1**, etc.

Using drivers depending on native API (Type 2 JDBC driver)

The JDBC type 2 driver, also known as the Native-API driver, is a database driver implementation that uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. For example: Oracle OCI driver is a type 2 driver.

These drivers often require additional components, such as DLL's or dynamic libraries, to be installed in addition to the driver JAR files. As mentioned earlier using a Type 2 driver is not recommended in DbVisualizer as pure java (Type 4) drivers often exists. An example is the Oracle Thin JDBC driver (Type 4) which is prefered over the Oracle OCI driver (Type 2).

Adding and using a Type 2 driver in DbVisualizer often involves driver dependant configuration tasks outside DbVisualizer. In some cases this is though not required as LIBRARY files can be directly configured for the driver. This is of course optional and configuration of libraries outside of DbVisualizer is always possible (using LD_LIBRARY_PATH or other environment configuration required by the driver).

Some examples follows

SSO or Windows Authentication using JTDS

In this case the driver template includes references to the needed LIBRARY file which is used in runtime by DbVisualizer. I.e no additional configuration is required by DbVisualizer.

SQL Server JDBC Driver

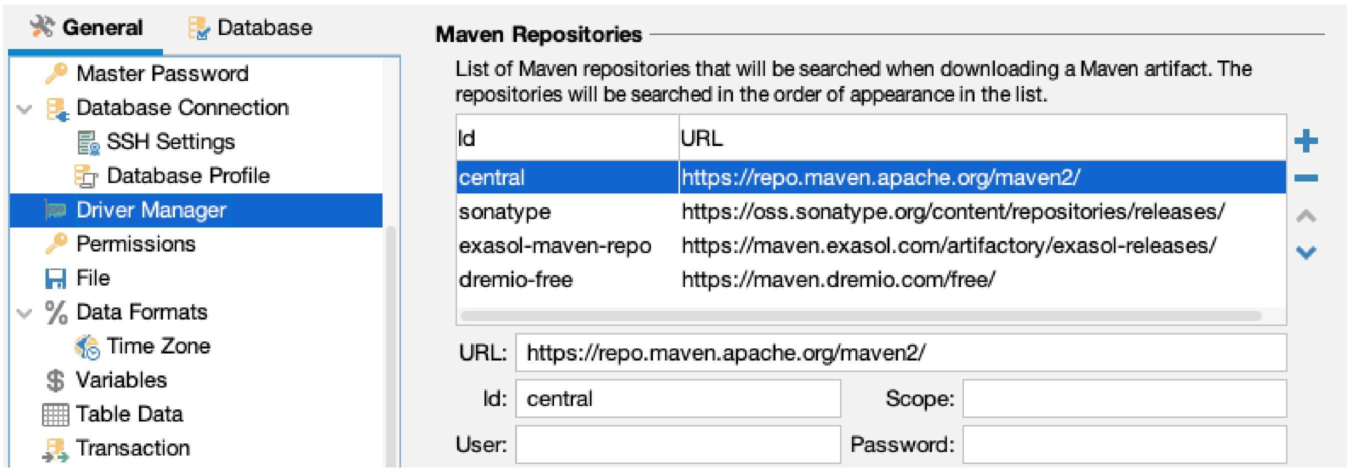
This driver definition also contains libraries (dll files) which is used by DbVisualizer without any need of additional configuration.

Changing drivers with LIBRARY artifacts require a DbVisualizer restart in order for the change to take effect.

Maven and Maven Repositories

Maven is a tool and framework for accessing and downloading 3rd-party libraries. Open source organisations and commercial companies provides JDBC driver libraries to on-line repositories that supports Maven for download. Many organisations also use internal Maven repositories. DbVisualizer comes with internal support for downloading with Maven in the Driver Manager. To download a driver using Maven you will need the **groupId**, **artifactId** and **version** for the driver, to be entered as a Maven artifact specification in the Driver Manager. In some cases you will also need an URL to a Maven repository including login credentials for the repository.

DbVisualizer comes with predefined templates containing Maven Artifacts in the Driver Manager and URLs for common Maven repositories.
You can add Maven repositories to search in in PreferencesDriver Manager.



Maven Repositories

List of Maven repositories that will be searched when downloading a Maven artifact. The repositories will be searched in the order of appearance in the list.

Id	URL
central	https://repo.maven.apache.org/maven2/
sonatype	https://oss.sonatype.org/content/repositories/releases/
exasol-maven-repo	https://maven.exasol.com/artifactory/exasol-releases/
dremio-free	https://maven.dremio.com/free/

URL: https://repo.maven.apache.org/maven2/

Id: central   Scope:

User: Password:

Repositories are searched from top down. Order can be changed and you can add and remove repositories. The scope is used to limit the search.