IBM Db2 Java Reactive Driver

# Installation Guide

Version 1.1.0

1-24-2023

# Contents

## 1. Getting the reactive driver

The Java Reactive Database Driver for Db2 is available in Maven Central repository. If you are using Maven, use the following dependency in your project pom.xml file. Then follow the instructions in the section on SWIDtag deployment.

```
<dependency>
    <groupId>com.ibm.db2</groupId>
    <artifactId>db2-r2dbc</artifactId>
    <version>1.1.0</version>
</dependency>
```

If you do not use Maven, see the sections below to download the reactive driver jar and its dependencies and how to setup your development environment.

## 2. Driver Documentation

The documentation for the driver is available in the following GitHub repository.

https://github.com/ibmdb/java_reactive_driver

This repository contains the following documents:

1. Installation Guide
2. Developer Guide
3. Debugging Guide
4. Javadoc API Documentation

This repository has samples and functional tests for the driver. It also has scripts that can help you set up your development environment, when not using Maven.

## 3. Setting the environment

For projects using Maven, you just need to include the above-mentioned driver dependency in your pom.xml file. You are all done with that.

If you do not use Maven, clone the following GitHub project:

```
git clone https://github.com/ibmdb/java_reactive_driver.git
```

Let us call this directory as installation directory. Add the `bin` folder in the installation directory to your PATH environment variable. If you are on Windows add the `bin/win` folder to your PATH environment variable.

The bin folder in the installation directory has a shell script called get_driver. Invoke bin/get_driver <version> from the installation directory. Here the <version> refers to driver version number, say for example 1.1.0. This will download the reactive driver jar and the Javadoc jar from Maven Central to the lib folder. This will also download SWIDtag file automatically (described below in the section on SWIDtag deployment). If you are on Windows, invoke bin\win\get_driver batch file.

Now make sure you have Maven installed in your system. Maven can be downloaded from http://maven.apache.org/. From a command window, make sure you can invoke mvn command.

To get the dependencies, from a command window, invoke bin/get_extlib command from the installation directory. If you are on Windows invoke bin\win\get_extlib command. Now all the jars that this driver depends on will be downloaded to extlib folder in the installation directory.

The driver jar in the lib folder, and all the jars in extlib require to be added to the CLASSPATH. For this, you can source the bash script bin/env from the installation directory. If you are on Windows invoke bin\env batch file. Your command window is now all set.

From the command window, run the following command:

```
java com.ibm.db2.r2dbc.Version
```

This will print the version of the driver you have. This command execution will also serve as a sanity test for your environment setup.

## 4. SWIDtag Deployment

SWIDtag is a file that requires to be deployed in the installation directory of the driver. This file is used by IBM License Manager Tool (ILMT). If you used the provided scripts to download the driver, SWIDtag file will be automatically downloaded and deployed in swidtag directory in the installation directory of the driver.

If you used maven, SWIDtag file will not be deployed automatically. You must download this file manually and deploy it in a directory by name swidtag in your home directory. The command given below uses curl to download this file.

```
curl -O https://repo1.maven.org/maven2/com/ibm/db2/db2-
r2dbc/VERSION/ibm.com_IBM_Db2_Java_Reactive_Driver-VERSION.swidtag
```

In the above command replace the tag VERSION with actual version number of the driver you downloaded, for example this can be 1.1.0. Change to swidtag directory and then invoke the above command. This will download and deploy the correct version of the SWIDtag file corresponding to the version of the driver you use.

## 5. Running samples

There are three samples in the GitHub repository. They are:

1. SampleApp.java
   This sample demonstrates all the CRUD operations – create, insert, update, select, delete, drop statements using the reactive driver. It also demonstrates how transactions and connection pool can be used.

2. BlobSample.java
   This sample demonstrates how Blob data in a Db2 database table can be read and updated by an application. It shows how the Blob values can be streamed to and from the server using the reactive driver.

3. ClobSample.java
   This sample demonstrates how Clob data in a Db2 database table can be read and updated by an application. Its shows how the text values can be streamed to and from the server using the reactive driver.

Before running the samples, update the `config.properties` file with the Db2 database connection details. Add the directory containing `config.properties` and `logback.xml` to your `classpath`. These two files are present under the `config` folder in the GitHub repository.

If you are running the samples from a command window, you can use the `build`, `run`, and `clean` scripts in the `bin` folder. For Windows, these scripts are under `bin\win` folder. From the `samples` folder invoke:

- `build` to build the samples
- `run [ SampleApp | BlobSample | ClobSample ]` to run the corresponding sample
- `clean` to clean the generated class files and log file.

## 6. Running unit tests

The GitHub repository also contains tests to test each of the functionality supported by the reactive driver. These tests are present under the `tests` folder and in the package `com.ibm.db2.r2dbc`.

Before running the tests, update the `config.properties` file with the Db2 database connection details. Add the directory containing `config.properties` and `logback.xml` to your `classpath`. These two files are present under the `config` folder in the GitHub repository. Also add the `tests` and `tests/resources` folder to your `classpath`.

These tests do create tables with the prefix “UNIT_TEST_” in the specified database and delete them after the tests are done. If there are any test errors, the errored tests may leave these tables undeleted. Hence, it is advisable to use an independent database to run these tests.

If you are running the tests from a command window, you can use the `build_test`, `run_test`, and `clean_test` scripts in the `bin` folder. For Windows, these scripts are under `bin\win` folder. From the `tests` folder invoke:

- `build_test` to build the tests.
- `run_test` to run all the tests.
- `clean_test` to clean the generated class files and log file.