ManageDbp (“**Manage** **D**ata**B**ase **P**rocesses”) is a server to manage, schedule, and run database process automation scripts. It exposes a RESTful web service interface to control its operation.

As a web server, ManageDbp can also serve static web content. In this way, static content that invokes the ManageDbp web services can be deployed.

This document tells how to configure and use the ManageDbp program.

# Prerequisites

ManageDbp is implemented in Java as a runnable JAR file. It uses the Dropwizard web services server framework. You must have a Java 8 JVM installed on your machine.

The ManageDbp JAR contains all the classes needed to run the program, including classes to connect to the MySQL database and the H2 database, the latter of which is embedded into the server. If using another database, you must have the JDBC JAR available.

If you are using MS SQL Server integrated security, you must download sqljdbc\_auth.dll from Microsoft.

The Dropwizard server configuration is controlled by a yaml file as described later in this document.

The DBPA scripting configuration is controlled by properties files as described later in this document. You should select a directory where you will place the properties files. This is your DBPA\_HOME directory. Set the DBPA\_HOME environment variable either globally or in a script that invokes the ManageDbp program.

# Syntax

ManageDbp should be run as a background process or service. Use your operating system facilities or a third-party solution to run it as such. For testing, it can be run as a normal process.

The class containing the main method is com.hauldata.dbpa.ManageDbp.

Invoke the program as follows:

java -cp *classPath* com.hauldata.dbpa.ManageDbp server *yamlFile*

The *classPath* must include the locations of the following:

1. the ManageDbp JAR
2. JDBC JARs used for databases other than MySQL and H2
3. the root directory under which an assets directory is found if serving static web content. The static content must be placed in the assets directory.

ManageDbp includes a guard to prevent multiple instances from running against the same database schema. If the program is terminated abnormally, the guard will be left in place, aborting any subsequent ManageDbp runs until the guard is reset. Reset it by running the program as follows:

java -cp *classPath* com.hauldata.dbpa.ManageDbp reset

When the program is invoked this way, it will reset the guard if needed and will then exit immediately. Subsequent ManageDbp runs using the “server yamlFile” command should then run normally.

# yaml File

Because it uses the Dropwizard framework, server configuration is controlled by a yaml file. The recommended file name is dbpa-manage.yaml. See the Dropwizard documentation for general guidance on setting up the file, e.g., at:

<https://www.dropwizard.io/1.3.5/docs/manual/configuration.html#man-configuration>

<https://github.com/dropwizard/dropwizard/blob/master/dropwizard-example/example.yml>

Details specific to ManageDbp are as follows.

In the server section, you must include the following:

rootPath: /api/

This indicates the RESTful services API is exposed under the api path of the root URL for the server.

If the RESTFUL services will be called from web pages served by a different server (not this ManageDbp instance), you must enable CORS (Cross-Origin Resource Sharing) by adding the following:

cors:

enabled: true

If finer CORS control Is required, in addition to enabled, the following parameters can be set: allowedOrigins, allowedHeaders, allowedMethods.

Database connectivity is not configured through the yaml file. Use a properties file as described below.

# Properties Files

The properties file contents all follow Java conventions. In particular, the backslash is an escape character. Therefore, to use a literal backslash character in a file path in a properties file, you must specify double backslashes. But note that you can use forward slashes for file path separators even when running under Microsoft Windows.

ManageDbp.\*.properties files control the operation of ManageDbp. However, when ManageDbp runs a job and invokes a DBPA script, script execution is controlled by RunDbp.\*.properties files. See *DBPA RunDbp User Guide* for guidance on those properties.

In most cases, if a ManageDbp.\*.properties file is not present, properties are read instead from the corresponding RunDbp.\*.properties file. See details below.

Logging from ManageDbp is controlled by the yaml file, not a properties file. You should *not* deploy a file named ManageDbp.log.properties.

## ManageDbp.jdbc.properties

This file configures the JDBC connection to the database containing your ManageDbp job information.

If present, properties are as follows:

driver=*driverJavaClass*

url=*databaseConnectionURL*

managerTablePrefix=*optionalPrefix*

driver and url are required properties. If this file is not present or a required property is missing, the properties in RunDbp.jdbc.properties are used instead. Note database containing your ManageDbp job information can be distinct from the database containing the business data accessed by your DBPA scripts.

If specified, managerTablePrefix is prefixed to each base table name when the ManageDbp database tables are accessed. The base tables have generic names such as Job, Schedule, etc. The prefix can be used to avoid naming collisions with existing tables. If the managerTablePrefix property is not specified, no prefix is added to the table names.

See the documentation for your database to determine the values to use for the driver and url properties. All other properties in the file are passed through to the driver itself. Again, see the documentation for your database to determine what other properties are required by the driver. Typically, at least the following are required:

user=*userName*

password=*userPassword*

Suggested properties to connect to a MySQL database running on your local machine are as follows:

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://localhost/*yourDatabaseName*

allowMultiQueries=true

user=*yourUser*

password=*yourPassword*

For MS SQL Server without integrated security, suggested properties are as follows:

driver=com.microsoft.sqlserver.jdbc.SQLServerDriver

url=jdbc:sqlserver://*ipAddressOrServerName*

user=*yourUser*

password=*yourPassword*

For MS SQL Server using integrated security you must set the integratedSecurity property true. Suggested properties to connect to an MS SQL Server database using integrated security are as follows:

driver=com.microsoft.sqlserver.jdbc.SQLServerDriver

url=jdbc:sqlserver://*ipAddressOrServerName*

integratedSecurity=true

## ManageDbp.mail.properties

This file configures the connection to your email server. Email is managed using the Java Mail API implemented in the javax.mail package. The connection is established using a password authenticator.

Email is used for sending alerts when job exception conditions occur. Properties as follows:

user=*yourMailUser*

password=*yourMailPassword*

alertFrom=*addressFromWhichAlertEmailsOriginate*

alertTo=*addressToWhichAlertEmailsAreSent*

alertSubject=*subjectLineOfAlertEmails*

The user, password, alertFrom, and alertTo properties are required. If this file is not present or a required property is missing, the properties in RunDbp.mail.properties are used instead. If a required property is not found in either file, no email alerts are sent.

The alertSubject property may contain the string %s which will be replaced by the script name in the actual subject line. If this setting is omitted, a default subject line is used which includes the script name.

All other properties in the file are passed through to the javax.mail.Session.getInstance( Properties, Authenticator) function using a password authenticator. See that function and documentation for your email server to determine what other properties are required.

To use the mail server for your gmail account, suggested properties are as follows:

user=*yourGmailAddress*

password=*yourGmailPassword*

mail.smtp.starttls.enable=true

mail.smtp.auth=true

mail.smtp.host=smtp.gmail.com

mail.smtp.port=587

## ManageDbp.path.properties

This file controls the directory paths where script files are found. If present, contents are as follows:

process=*pathToProcessScriptFiles*

Any other contents are ignored.

If this file is not present or the process property is missing, the property in RunDbp.path.properties is used instead.

In normal use, it is recommended that this file be omitted and the value from RunDbp.path.properties be used instead. See *DBPA RunDbp User Guide* for the use of the process setting.

## ManageDbp.statsd.properties

This file configures the connection to a statsd server for collecting statistics. If present, contents are as follows:

prefix=*prefix*

hostname=*hostname*

port=*port*

All properties are required. If any is missing, statistics are not sent to statsd.

*prefix* followed by a dot character is prepended to each statistic sent to statsd. *hostname* and *port* identify the location of the statsd server. All statics are counters. The statistics sent are as follows:

loadFailed a job failed to load

submitFailed a job loaded but failed when submitted for execution

runInProgress a job has started running

runFailed a job run ended with failure status

runSucceeded a job run ended with success status

runTerminated a job run was terminated by request

controllerShutdown a job run was terminated because ManageDbp was shut down

databaseError ManageDbp encountered an error accessing the database

alertError ManageDbp encountered an error sending an alert email

In normal use, runInProgress should be recorded regularly. runSucceeded should closely track runInProgress. runFailed should be rare. All others should not happen at all.