

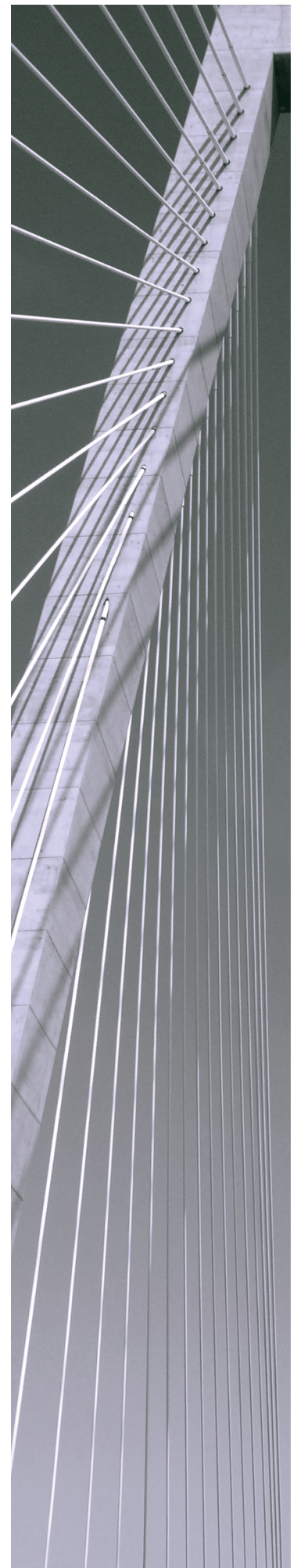


Simba Athena JDBC Driver with SQL Connector

Installation and Configuration Guide

Simba Technologies Inc.

Version 2.0.5
August 15, 2018



Copyright © 2018 Simba Technologies Inc. All Rights Reserved.

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. No part of this publication, or the software it describes, may be reproduced, transmitted, transcribed, stored in a retrieval system, decompiled, disassembled, reverse-engineered, or translated into any language in any form by any means for any purpose without the express written permission of Simba Technologies Inc.

Trademarks

Simba, the Simba logo, SimbaEngine, and Simba Technologies are registered trademarks of Simba Technologies Inc. in Canada, United States and/or other countries. All other trademarks and/or servicemarks are the property of their respective owners.

Contact Us

Simba Technologies Inc.
938 West 8th Avenue
Vancouver, BC Canada
V5Z 1E5

Tel: +1 (604) 633-0008

Fax: +1 (604) 633-0004

www.simba.com

About This Guide

Purpose

The *Simba Athena JDBC Driver with SQL Connector Installation and Configuration Guide* explains how to install and configure the Simba Athena JDBC Driver with SQL Connector on all supported platforms. The guide also provides details related to features of the driver.

Audience

The guide is intended for end users of the Simba Athena JDBC Driver.

Knowledge Prerequisites

To use the Simba Athena JDBC Driver, the following knowledge is helpful:

- Familiarity with the platform on which you are using the Simba Athena JDBC Driver
- Ability to use the data store to which the Simba Athena JDBC Driver is connecting
- An understanding of the role of JDBC technologies in connecting to a data store
- Experience creating and configuring JDBC connections
- Exposure to SQL

Document Conventions

Italics are used when referring to book and document titles.

Bold is used in procedures for graphical user interface elements that a user clicks and text that a user types.

Monospace font indicates commands, source code or contents of text files.

Note:

A text box with a pencil icon indicates a short note appended to a paragraph.

! Important:

A text box with an exclamation mark indicates an important comment related to the preceding paragraph.

Table of Contents

About the Simba Athena JDBC Driver	7
About Amazon Athena	7
About the Driver	7
System Requirements	9
Simba Athena JDBC Driver Files	10
Installing and Using the Simba Athena JDBC Driver	11
Referencing the JDBC Driver Libraries	11
Registering the Driver Class	12
Building the Connection URL	13
Example: Using the Driver in SQL Workbench	14
Examples: Using the Driver in a Java Application	20
Configuring Authentication	27
Using IAM Credentials	27
Using the AWSCredentialsProvider Interface	27
Configuring Query Result Encryption	31
Configuring Proxy Connections	32
Configuring Logging	34
Features	36
Catalog and Schema Support	36
File Formats	36
Fetch Size	36
Data Types	37
Integration with AWS Glue	39
Security and Authentication	39
Driver Configuration Options	41
AwsCredentialsProviderArguments	41
AwsCredentialsProviderClass	42
AwsRegion	42
BinaryColumnLength	43
ComplexTypeColumnLength	43
ConnectionTest	43

Installation and Configuration Guide

ConnectTimeout	44
LogLevel	44
LogPath	45
MaxCatalogNameLength	46
MaxColumnNameLength	46
MaxErrorRetry	46
MaxSchemaNameLength	47
MaxTableNameLength	47
MetadataRetrievalMethod	47
NonProxyHosts	48
PWD	49
PreemptiveBasicProxyAuth	49
ProxyDomain	49
ProxyHost	50
ProxyPort	50
ProxyPWD	50
ProxyUID	51
ProxyWorkstation	51
RowsToFetchPerBlock	51
S3OutputEncKMSKey	52
S3OutputEncOption	52
S3OutputLocation	53
Schema	53
SocketTimeout	54
StringColumnLength	54
UseArraySupport	55
OUseAwsLogger	55
UseResultsetStreaming	56
UID	56
Appendix: Migrating to Later Driver Versions	57
Upgrading From 1.x to 2.0.x	57
Upgrading from 2.0.2 to 2.0.5 or later	63
Third-Party Trademarks	64
Third-Party Licenses	65

About the Simba Athena JDBC Driver

About Amazon Athena

Amazon Athena is a serverless interactive query service capable of querying data from Amazon Simple Storage Service (S3) using SQL. It is designed for short, interactive queries that are useful for data exploration. Athena enables you to run ad-hoc queries and quickly analyze data that is stored in S3 without ETL processes. Query results are stored in an S3 bucket and made available for analysis in BI tools.

The data formats that Athena supports include CSV, JSON, Parquet, Avro, and ORC. Unlike traditional RDBMS or SQL-on-Hadoop solutions that require centralized schema definitions, Athena can query self-describing data as well as complex or multi-structured data that is commonly seen in big data systems. Moreover, Athena does not require a fully structured schema and can support semi-structured or nested data types such as JSON.

Amazon Athena processes the data in record batches and discovers the schema during the processing of each record batch. Thus, Athena has the capability to support changing schemas over the lifetime of a query. Athena reconfigures its operators and handles these situations to ensure that data is not lost.

Note:

- Access from Athena to your S3 data store is configured through Amazon Web Services (AWS). For information about enabling Athena to access S3 data stores, see the Amazon Athena documentation: <http://docs.aws.amazon.com/athena/latest/ug/what-is.html>.
- When using Athena, you are charged for each query that you run. The amount that you are charged is based on the amount of data scanned by the query. For more information, see *Amazon Athena Pricing*: <https://aws.amazon.com/athena/pricing/>.

About the Driver

The Simba Athena JDBC Driver enables organizations to connect their BI tools to the Amazon Athena query service, enabling Business Intelligence, analytics, and reporting on the data that Athena returns from Amazon S3 databases. If the AWS Glue service is available in the region and Athena has been migrated to use AWS Glue to manage the data catalog, then the driver retrieves catalog metadata via the AWS Glue service. Otherwise, the driver retrieves catalog metadata from the Athena-managed data catalog.

The Simba Athena JDBC Driver complies with the JDBC 4.1 and 4.2 data standards. JDBC is one of the most established and widely supported APIs for connecting to and working with databases. At the heart of the technology is the JDBC driver, which connects an application to the database. For more information about JDBC, see *Data Access Standards* on the Simba Technologies website: <https://www.simba.com/resources/data-access-standards-glossary>.

The *Simba Athena JDBC Driver with SQL Connector Installation and Configuration Guide* is suitable for users who are looking to access data returned by the Athena query service from their desktop environment. Application developers may also find the information helpful. Refer to your application for details on connecting via JDBC.

System Requirements

Each machine where you use the Simba Athena JDBC Driver must have Java Runtime Environment (JRE) 8.0 installed. If you are using the driver with JDBC API version 4.2, then you must use JRE 8.0.

Simba Athena JDBC Driver Files

The Simba Athena JDBC Driver is delivered in the ZIP archive `SimbaAthenaJDBC-[Version].zip`, where `[Version]` is the version number of the driver.

This archive contains the fat JARs for all of the JDBC API versions that are supported by the driver: JDBC 4.1 and 4.2. Each JAR contains all of the required third-party libraries and dependencies for the driver.

Installing and Using the Simba Athena JDBC Driver

To install the Simba Athena JDBC Driver on your machine, extract the appropriate JAR file from the ZIP archive to the directory of your choice.

! Important:

If you received a license file through email, then you must copy the file into the same directory as the driver JAR file before you can use the Simba Athena JDBC Driver.

To access the Athena service using the Simba Athena JDBC Driver, you need to configure the following:

- The list of driver library files (see [Referencing the JDBC Driver Libraries](#) on page 11)
- The `Driver` or `DataSource` class (see [Registering the Driver Class](#) on page 12)
- The connection URL for the driver (see [Building the Connection URL](#) on page 13)

You can use the Simba Athena JDBC Driver in a JDBC application or a Java application.

- For an example workflow that demonstrates how to use the driver in a JDBC application, see [Example: Using the Driver in SQL Workbench](#) on page 14.
- For code examples that demonstrate how to use the driver in a Java application, see [Examples: Using the Driver in a Java Application](#) on page 20.

Referencing the JDBC Driver Libraries

Before you use the Simba Athena JDBC Driver, the JDBC application or Java code that you are using to connect to your data must be able to access the driver JAR file. In the application or code, specify the appropriate fat JAR file for the JDBC version that you are using.

Using the Driver in a JDBC Application

Most JDBC applications provide a set of configuration options for adding a list of driver library files. Use the provided options to include the appropriate fat JAR file from the ZIP archive as part of the driver configuration in the application. For more information, see the documentation for your JDBC application.

Using the Driver in Java Code

You must include all the driver library files in the class path. This is the path that the Java Runtime Environment searches for classes and other resource files. For more information, see "Setting the Class Path" in the appropriate Java SE Documentation.

For Java SE 7:

- For Windows:
<http://docs.oracle.com/javase/7/docs/technotes/tools/windows/classpath.html>
- For Linux and Solaris:
<http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/classpath.html>

For Java SE 8:

- For Windows:
<http://docs.oracle.com/javase/8/docs/technotes/tools/windows/classpath.html>
- For Linux and Solaris:
<http://docs.oracle.com/javase/8/docs/technotes/tools/solaris/classpath.html>

Registering the Driver Class

Before connecting to your data, you must register the appropriate class for your application.

The following is a list of the classes used to connect the Simba Athena JDBC Driver to the Athena service. The `Driver` classes extend `java.sql.Driver`, and the `DataSource` classes extend `javax.sql.DataSource` and `javax.sql.ConnectionPoolDataSource`.

The driver supports the following fully-qualified class names (FQCNs) that are independent of the JDBC version:

- `com.simba.athena.jdbc.Driver`
- `com.simba.athena.jdbc.DataSource`

The following sample code shows how to use the `DriverManager` to establish a connection for JDBC:

```
private static Connection connectViaDM() throws Exception
{
    Connection connection = null;
    Class.forName(DRIVER_CLASS);
    connection = DriverManager.getConnection(CONNECTION_URL);
    return connection;
}
```

```
}
```

The following sample code shows how to use the `DataSource` class to establish a connection:

```
private static Connection connectViaDS() throws Exception
{
    Connection connection = null;
    Class.forName(DRIVER_CLASS);
    DataSource ds = new com.simba.athena.jdbc.DataSource();
    ds.setURL(CONNECTION_URL);
    connection = ds.getConnection();
    return connection;
}
```

Building the Connection URL

Use the connection URL to supply connection information to the data store that you are accessing.

Standard connection string


The following is the format of the connection URL for the Simba Athena JDBC Driver:

```
jdbc:awsathena://AwsRegion=[Region];UID=[AccessKey];PWD=[SecretKey];S3OutputLocation=[Output];[Property1]=[Value1];[Property2]=[Value2];...
```

Using an endpoint URL

The following is the format of a connection URL using an endpoint.

```
jdbc:awsathena://athena.[Region].amazonaws.com:443;UID=[AccessKey];PWD=[SecretKey];S3OutputLocation=[Output];[Property1]=[Value1];[Property2]=[Value2];...
```

 **Note:**

If both `AwsRegion` and `endpoint` are present the `AWSRegion` takes precedence.

The variables are defined as follows:

- *[Region]* is the AWS region of the Athena instance that you want to connect to.
- *[AccessKey]* is the access key provided by your AWS account.
- *[SecretKey]* is the secret key provided by your AWS account.
- *[Output]* is the path of the Amazon S3 location where you want to store query results, prefixed by `s3://`.
- *[Property1..N]* and *[Value1..N]* are additional connection properties supported by the driver. For a list of the properties available in the driver, see [Driver Configuration Options](#) on page 41.

! Important:

- Properties are case-sensitive.
- Do not duplicate properties in the connection URL.

Example: Using the Driver in SQL Workbench

SQL Workbench is one of many applications that use drivers to query and view data. The instructions below provide general guidelines for configuring and using the Simba Athena JDBC Driver in SQL Workbench.

Before You Begin

Before you can use the driver in SQL Workbench, you must do the following:

- Download and install SQL Workbench. You can download the application from <http://www.sql-workbench.net/downloads.html>.
- Download and extract the driver ZIP archive (`SimbaAthenaJDBC-[Version].zip`) into the SQL Workbench directory.
- Set up the Athena service. For more information, see "Setting Up" in the Amazon Athena Documentation: <http://docs.aws.amazon.com/athena/latest/ug/setting-up.html>.

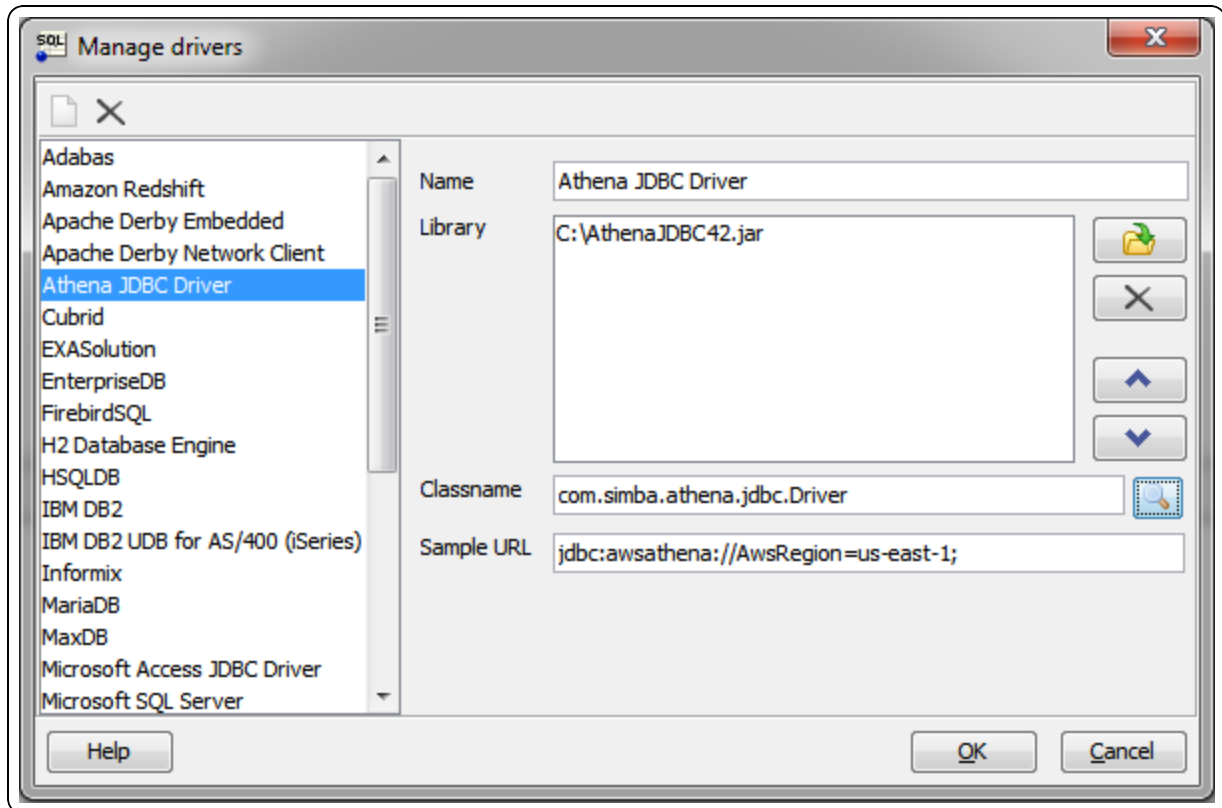
Configuring SQL Workbench to Use the Driver

Add the Simba Athena JDBC Driver to the list of drivers in SQL Workbench, and then create a connection profile that contains the necessary connection information.

To configure SQL Workbench to use the driver:

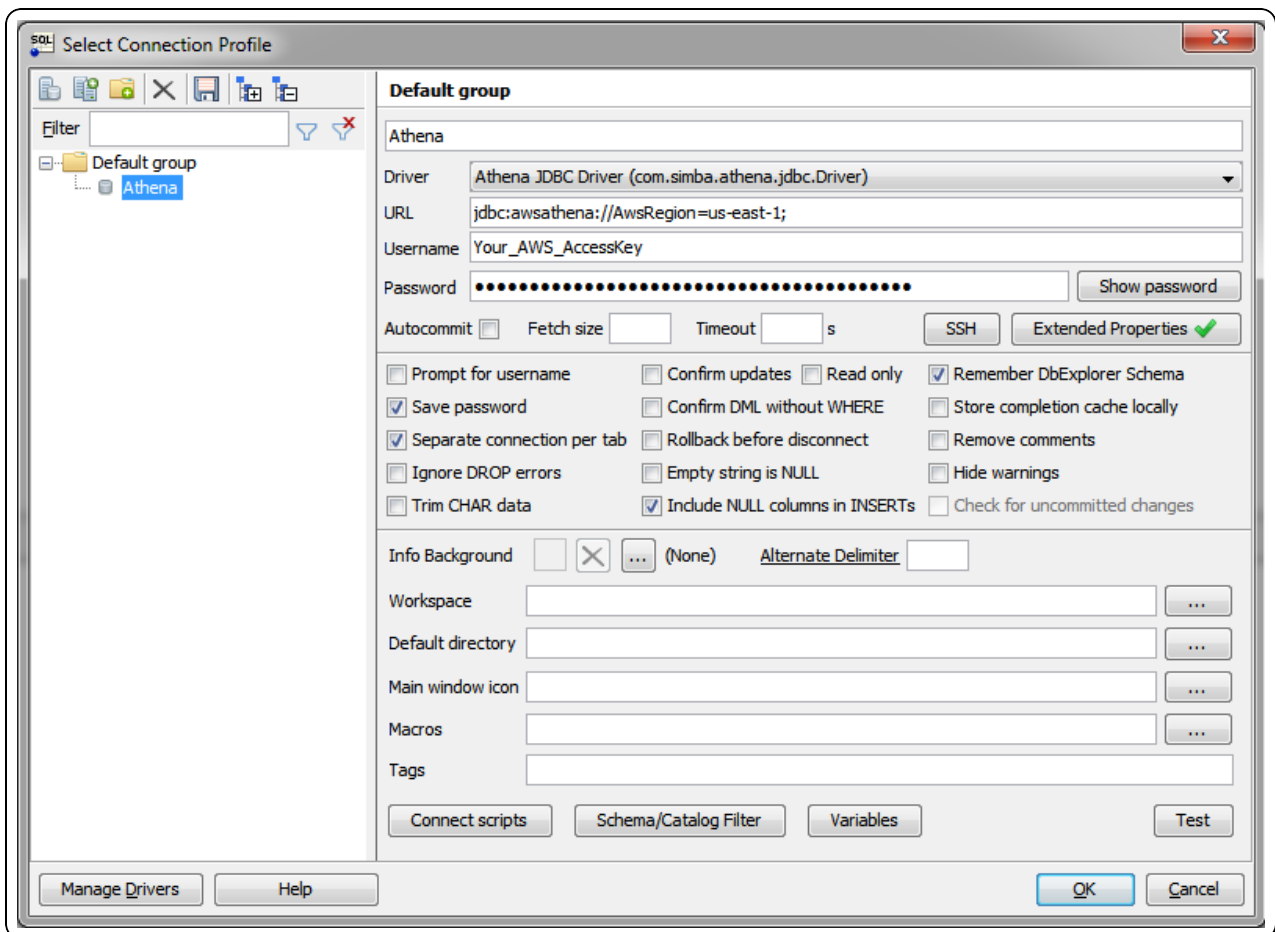
1. In SQL Workbench, select **File > Manage Drivers**.
2. In the Manage Drivers dialog box, specify the following values in the fields:

Field Name	Value
Name	A name that you want to use to identify the Simba Athena JDBC Driver in SQL Workbench. For example, Athena JDBC Driver .
Library	The full path and name of the <code>AthenaJDBC [APIVersion].jar</code> file, where <code>[APIVersion]</code> is the JDBC version number that the driver supports. For example, AthenaJDBC42.jar for the driver that supports JDBC 4.2.
Classname	com.simba.athena.jdbc.Driver
Sample URL	A connection URL that only specifies the AWS region of the Athena instance that you want to connect to, using the format <code>jdbc:awsathena://AwsRegion=[Region];</code> For example, jdbc:awsathena://AwsRegion=us-east-1;



3. Click **OK** to save your settings and close the Manage Drivers dialog box.
4. Click **File > Connect Window**.
5. In the Select Connection Profile dialog box, create a new connection profile named "Athena".
6. From the **Driver** drop-down list, select the driver that you configured in step 2. The driver is listed with the name that you specified in step 2, followed by the classname.
7. To specify required connection information, specify the following values in the fields:

Field Name	Value
URL	<p>A connection URL that only specifies the AWS region of the Athena instance that you want to connect to, using the format</p> <pre>jdbc:awsathena://AwsRegion=[Region];.</pre> <p>For example, <code>jdbc:awsathena://AwsRegion=us-east-1;</code></p> <p>By default, this field is automatically populated with the Sample URL value that you specified for the selected driver.</p>
Username	The access key provided by your AWS account.
Password	The secret key provided by your AWS account.



8. Click **Extended Properties**, and add a property named `S3OutputLocation`. Set the value of this property to the path of the Amazon S3 location where you want to store query results, prefixed by `s3://`.

For example, to store Athena query results in a folder named "test-folder-1" inside an S3 bucket named "query-results-bucket", you would set the `S3OutputLocation` property to `s3://query-results-bucket/test-folder-1`.

9. Click **OK** to save your settings and close the Edit Extended Properties dialog box.
10. Click **OK** to save your connection profile and close the Select Connection Profile dialog box.

You can now use the Simba Athena JDBC Driver in SQL Workbench to query and view data.

Querying Data with SQL Workbench

Use the Statement window in SQL Workbench to execute queries on your data. You can also execute CREATE statements to add new tables, and create and use custom databases.

Note:

By default, the driver queries the default database. To distinguish between tables in the default and custom databases, when writing your queries, use the database identifier as a namespace prefix to your table name.

To query data with SQL Workbench:

1. In the Statement window, type a query that creates a table in the default database. For example:

```
CREATE EXTERNAL TABLE IF NOT EXISTS integer_table (  
  KeyColumn STRING,  
  Column1 INT)  
ROW FORMAT SERDE  
  'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'  
WITH SERDEPROPERTIES ('serialization.format' = ',',  
  'field.delim' = ',')  
LOCATION 's3://athena-examples/integer_table/'
```

2. Click **Execute**.
3. Run a simple query to retrieve some data, and then view the results. For

example:

```
SELECT * FROM integer_table
```

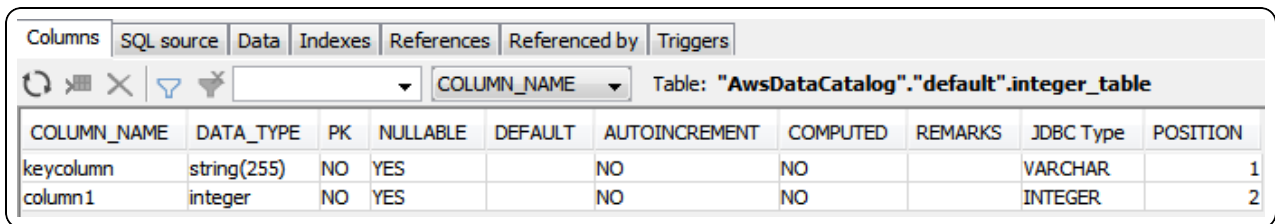
You can now view details about the retrieved data in the Data Explorer tab, as described below.

Exploring Data with SQL Workbench

Use the Data Explorer tab to view details about your retrieved data.

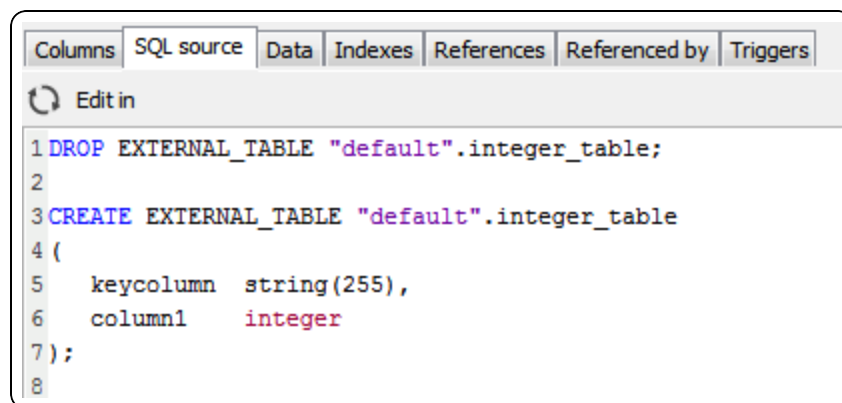
To explore data with SQL Workbench:

1. Select the **Data Explorer** tab, and then select the default schema (or database).
2. Select the **integer_table** table. SQL Workbench loads the Columns tab, which shows the table schema.



COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DEFAULT	AUTOINCREMENT	COMPUTED	REMARKS	JDBC Type	POSITION
keycolumn	string(255)	NO	YES		NO	NO		VARCHAR	1
column1	integer	NO	YES		NO	NO		INTEGER	2

3. Select the other tabs to view more information about the integer_table table. For example:
 - Select the **SQL Source** tab to view the queries that were used to generate the table.



```

1 DROP EXTERNAL_TABLE "default".integer_table;
2
3 CREATE EXTERNAL_TABLE "default".integer_table
4 (
5   keycolumn string(255),
6   column1 integer
7 );
8

```

- Select the **Data** tab to view a list of the rows returned from the table.

keycolumn	column1
Null	
Zero	0
One	1
MinusOne	-1
Two	2
MaxTInt	127
MinTInt	-128
MaxUTInt	255
MaxTIntP1	128
MinTIntM1	-129
MaxUTIntP1	256
MaxSInt	32767
MinSInt	-32768
MaxUSInt	65535
MinSIntM1	-32769
MaxSIntP1	32768
MaxUSIntP1	65536
MaxInt	2147483647
MinInt	-2147483648

You can repeat the procedures described above to retrieve and explore different data using the Simba Athena JDBC Driver in SQL Workbench.

Examples: Using the Driver in a Java Application

The following code examples demonstrate how to use the Simba Athena JDBC Driver in a Java application:

- [Example: Creating a Driver](#) on page 20
- [Examples: Using a Credentials Provider](#) on page 21
- [Example: Executing a SELECT Query](#) on page 23
- [Example: Running a CREATE Statement](#) on page 24
- [Example: Listing Tables](#) on page 24

Example: Creating a Driver

This example demonstrates how to create an instance of the Simba Athena JDBC Driver in a Java application:

```
Properties info = new Properties();
info.put("UID", "AWSAccessKey");
info.put("PWD", "AWSSecretAccessKey");
```

```
info.put("S3OutputLocation", "s3://my-athena-result-  
bucket/test/");  
  
Class.forName("com.simba.athena.jdbc.Driver");  
  
Connection connection = DriverManager.getConnection  
("jdbc:awsathena://AwsRegion=us-east-1;", info);
```

Examples: Using a Credentials Provider

The following examples demonstrate different ways of using a credentials provider that implements the `AWSCredentialsProvider` interface with the JDBC driver:

- [Example: DefaultAWSCredentialsProviderChain](#) on page 21
- [Example: PropertiesFileCredentialsProvider](#) on page 21
- [Example: InstanceProfileCredentialsProvider](#) on page 22
- [Example: CustomSessionCredentialsProvider](#) on page 22

For more information about configuring the driver to authenticate your connection using a credentials provider, see [Using the AWSCredentialsProvider Interface](#) on page 27.

Example: DefaultAWSCredentialsProviderChain

This example demonstrates how to use the `DefaultAWSCredentialsProviderChain`. You do not need to supply any credential provider arguments because they are taken from one of the locations in the default credentials provider chain. For detailed information about configuring default credentials, see "Using the Default Credential Provider Chain" in the *AWS SDK for Java Developer Guide*: <http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html#credentials-default>.

```
Properties info = new Properties();  
info.put("AwsCredentialsProviderClass",  
"com.simba.athena.amazonaws.auth.DefaultAWSCredentialsProvid  
erChain");
```

Example: PropertiesFileCredentialsProvider

This example demonstrates how to use the `PropertiesFileCredentialsProvider`, which uses only one argument and obtains the required credentials from a file:

```
Properties info = new Properties();  
info.put("AwsCredentialsProviderClass",
```

```
"com.simba.athena.amazonaws.auth.PropertiesFileCredentialsProvider");
info.put("AwsCredentialsProviderArguments",
"/Users/myUser/.athenaCredentials");
```

With the implementation shown above, the credentials provider obtains the required credentials from a file named `/Users/myUser/.athenaCredentials`, which should contain the following text:

```
accessKey=[YourAccessKey]
secretKey=[YourSecretKey]
```

The variables are defined as follows:

- `[YourAccessKey]` is the access key provided by your AWS account.
- `[YourSecretKey]` is the secret key provided by your AWS account.

Example: InstanceProfileCredentialsProvider

This example demonstrates how to use the `InstanceProfileCredentialsProvider`. You do not need to supply any credential provider arguments because they are provided using the EC2 instance profile for the instance on which you are running your application. However, you still need to set the `AwsCredentialsProviderClass` property to this class name.

```
Properties info = new Properties();
info.put("AwsCredentialsProviderClass",
"com.simba.athena.amazonaws.auth.InstanceProfileCredentialsProvider");
```

Example: CustomSessionCredentialsProvider

`CustomSessionsCredentialsProvider` is not included with the driver, so you must create it before you can use it.

This example demonstrates how to create a custom credentials provider named `CustomSessionCredentialsProvider` that uses an access key, secret key, and session token:

```
package com.example;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSCredentialsProvider;
```

```
import
com.simba.athena.amazonaws.auth.BasicSessionCredentials;

public class CustomSessionCredentialsProvider implements
AWSCredentialsProvider
{
    private BasicSessionCredentials m_credentials;

    public CustomSessionCredentialsProvider(
        String awsAccessKey,
        String awsSecretKey,
        String sessionToken)
    {
        m_credentials =
            new BasicSessionCredentials(
                awsAccessKey,
                awsSecretKey,
                sessionToken);
    }

    @Override
    public AWSCredentials getCredentials()
    {
        return m_credentials;
    }

    @Override
    public void refresh(){}
}
```

The following example demonstrates how to use CustomSessionCredentialsProvider after it has been created:

```
Properties info = new Properties();
info.put("AwsCredentialsProviderClass",
"com.example.CustomSessionCredentialsProvider");
String providerArgs = "My_Access_Key," + "My_Secret_Key," +
"My_Token";
info.put("AwsCredentialsProviderArguments", providerArgs);
```

Example: Executing a SELECT Query

This example demonstrates how to execute a SELECT query:

```
Statement statement = connection.createStatement();
ResultSet queryResults = statement.executeQuery("SELECT *
FROM integer_table");
```

Example: Running a CREATE Statement

This example demonstrates how to run a CREATE statement:

```
Statement statement = connection.createStatement();
ResultSet queryResults = statement.executeQuery("CREATE
EXTERNAL TABLE IF NOT EXISTS tableName (Col1 String)
LOCATION 's3://bucket/tableLocation'");
```

Example: Listing Tables

This example demonstrates how to list the tables from the result set of a query:

```
import java.sql.*;
import java.util.Properties;

public class AthenaJDBCDemo {

    static final String athenaUrl =
        "jdbc:awsathena://AwsRegion=us-east-1;";

    public static void main(String[] args) {

        Connection conn = null;
        Statement statement = null;

        try {
            Class.forName("com.simba.athena.jdbc.Driver");
            Properties info = new Properties();
            info.put("S3OutputLocation", "s3://my-athena-
result-bucket/test/");
            info.put("LogPath", "/Users/myUser/athenaLog");
            info.put("LogLevel", "6");
            info.put
                ("AwsCredentialsProviderClass", "com.simba.athena.am
azonaws.auth.PropertiesFileCredentialsProvider");
            info.put
```



```
    ("AwsCredentialsProviderArguments", "/Users/myUser/.  
athenaCredentials");  
    String databaseName = "default";  
  
    System.out.println("Connecting to Athena...");  
    conn = DriverManager.getConnection(athenaUrl,  
    info);  
  
    System.out.println("Listing tables...");  
    String sql = "show tables in "+ databaseName;  
    statement = conn.createStatement();  
    ResultSet rs = statement.executeQuery(sql);  
  
    while (rs.next()) {  
        //Retrieve table column.  
        String name = rs.getString("tab_name");  
  
        //Display values.  
        System.out.println("Name: " + name);  
    }  
    rs.close();  
    conn.close();  
} catch (Exception ex) {  
    ex.printStackTrace();  
} finally {  
    try {  
        if (statement != null)  
            statement.close();  
    } catch (Exception ex) {  
  
    }  
    try {  
        if (conn != null)  
            conn.close();  
    } catch (Exception ex) {  
  
        ex.printStackTrace();  
    }  
}  
System.out.println("Finished connectivity test.");  
}
```

```
}  
}
```

Configuring Authentication

To access data from Athena, you must authenticate the connection. You can configure the Simba Athena JDBC Driver to provide your credentials and authenticate the connection using one of the following methods:

- [Using IAM Credentials](#) on page 27
- [Using the AWSCredentialsProvider Interface](#) on page 27

Using IAM Credentials

You can configure the driver to authenticate the connection using an access key and a secret key that are specified directly in the connection information.

To configure authentication using IAM credentials:

1. Set the `UID` property to the access key provided by your AWS account.
2. Set the `PWD` property to the secret key provided by your AWS account.

Using the AWSCredentialsProvider Interface

You can configure the driver to authenticate the connection using a class that implements the `AWSCredentialsProvider` interface. For detailed information about this interface, see the Amazon AWS documentation for Interface `AWSCredentialsProvider`: <http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/auth/AWSCredentialsProvider.html>.

To configure authentication using the `AWSCredentialsProvider` interface:

1. Set the `AwsCredentialsProviderClass` property to a fully qualified class name that implements the `AWSCredentialsProvider` interface. This class can be an implementation from the AWS SDK, or a custom implementation.

! Important:

- If you use an implementation from the AWS SDK, you may use the shaded package name for `amazonaws` that is included inside the driver jar. This is `com.simba.athena.amazonaws`.
 - If you use a custom implementation, include that implementation in your class path.
2. If necessary, set the `AwsCredentialsProviderArguments` property to a comma-separated list of `String` arguments for the constructor of the `AwsCredentialsProviderClass`.

Be aware of the following restrictions:

- The driver only supports String arguments for the constructor parameters.
- Multiple arguments must be separated by a comma (,).
- Surrounding spaces are not included in the parsed arguments.
- To escape a single character, use a backslash (\) before that character. To indicate a backslash in an argument, use two backslashes (\\).
- To escape all commas in an argument, enclose the argument in quotation marks ("). To indicate a quotation mark in a quoted argument, use a backslash (\) before that quotation mark.

For more detailed instructions about how to configure authentication using various implementations of the `AWSCredentialsProvider` interface, see the following:

- [Using DefaultAWSCredentialsProviderChain](#) on page 28
- [Using PropertiesFileCredentialsProvider](#) on page 29
- [Using InstanceProfileCredentialsProvider](#) on page 29
- [Using a Custom Credentials Provider](#) on page 30

For code examples that demonstrate how to use each type of credentials provider in a Java application, see [Examples: Using the Driver in a Java Application](#) on page 20.

Using DefaultAWSCredentialsProviderChain

To configure authentication using DefaultAWSCredentialsProviderChain:

1. Set the `AwsCredentialsProviderClass` property to `com.simba.athena.amazonaws.auth.DefaultAWSCredentialsProviderChain`.
2. Do not set the `AwsCredentialsProviderArguments` property.

The arguments are taken from one of the locations in the default credentials provider chain. For detailed information about configuring default credentials, see "Using the Default Credential Provider Chain" in the *AWS SDK for Java Developer Guide*: <http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html#credentials-default>.

For a code example that demonstrates how to use the `DefaultAWSCredentialsProviderChain` in a Java application, see [Example: DefaultAWSCredentialsProviderChain](#) on page 21.

Using PropertiesFileCredentialsProvider

To configure authentication using PropertiesFileCredentialsProvider:

1. Create a text file called `athenaCredentials.props`. This file should contain the following text:

```
accessKey = [AccessKey]
secretKey = [SecretKey]
```

The variables are defined as follows:

- `[AccessKey]` is the access key provided by your AWS account.
 - `[SecretKey]` is the secret key provided by your AWS account.
2. Set the `AwsCredentialsProviderClass` property to `com.simba.athena.amazonaws.auth.PropertiesFileCredentialsProvider`.
 3. Set the `AwsCredentialsProviderArguments` property to the full path and filename of the `athenaCredentials.props` file. For example, `"/Users/skroob/athenaCredentials.props"`.

For a code example that demonstrates how to use the `PropertiesFileCredentialsProvider` in a Java application, see [Example: PropertiesFileCredentialsProvider](#) on page 21.

Using InstanceProfileCredentialsProvider

To configure authentication using InstanceProfileCredentialsProvider:

1. Set the `AwsCredentialsProviderClass` property to `com.simba.athena.amazonaws.auth.InstanceProfileCredentialsProvider`.
2. Do not set the `AwsCredentialsProviderArguments` property.

The arguments are provided by the EC2 instance profile for the instance on which you are running your application. For more detailed information about configuring `InstanceProfileCredentialsProvider`, see "IAM Roles for Amazon EC2" in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html>.

For a code example that demonstrates how to use the `InstanceProfileCredentialsProvider` in a Java application, see [Example: InstanceProfileCredentialsProvider](#) on page 22.

Using a Custom Credentials Provider

This example shows a custom credentials provider, `CustomSessionsCredentialsProvider`, that uses an access and secret key in addition to a session token. `CustomSessionsCredentialsProvider` is shown for example only and is not included in the driver. You must create custom providers before you can use them.

To configure authentication using a custom credentials provider:

1. Create a credentials provider called `CustomSessionsCredentialsProvider` that uses an access key, secret key, and session token for authentication.
2. In the connection URL, set the `AwsCredentialsProviderClass` property to `com.example.CustomSessionCredentialsProvider`.
3. Set the `AwsCredentialsProviderArguments` property to `"My_Access_Key, My_Secret_Key, My_Token"`.
4. Generate `My_Access_Key`, `My_Secret_Key` and `My_Token` using AWS Security Token Service. For detailed instructions, see "Temporary Security Credentials" in the *AWS Identity and Access Management User Guide*:
http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html.

For code examples that demonstrate how to create and use the `CustomSessionCredentialsProvider` in a Java application, see [Example: CustomSessionCredentialsProvider](#) on page 22.

To use a custom credential provider in an application that has a graphical user interface (GUI), start by exporting the implementation as a JAR file. Then, using the options in the application, include that JAR file along with the driver JAR files.

Configuring Query Result Encryption

You can configure the Simba Athena JDBC Driver to encrypt your query results using any of the encryption protocols that Athena supports.

To configure query result encryption:

1. Set the `S3OutputEncOption` property to one of the following values.

Option Name	Description
SSE_S3	The driver uses server-side encryption with an Amazon S3-managed key.
SSE_KMS	The driver uses server-side encryption with an AWS KMS-managed key.
CSE_KMS	The driver uses client-side encryption with an AWS KMS-managed key.

For detailed information about these encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*:

<http://docs.aws.amazon.com/athena/latest/ug/encryption.html>.

2. If you specified `SSE_KMS` or `CSE_KMS` in the previous step, then set the `S3OutputEncKMSKey` property to the KMS key ARN or ID to use for encrypting data.

Configuring Proxy Connections

You can configure the driver to connect through a proxy server instead of connecting directly to the Athena service. When connecting through a proxy server, the driver supports basic authentication and NTLM authentication.

To configure a proxy connection:

1. Set the `ProxyHost` property to the IP address or host name of your proxy server.
2. Set the `ProxyPort` property to the number of the TCP port that the proxy server uses to listen for client connections.
3. Optionally, to connect to certain hosts directly even when a proxy connection has been configured, set the `NonProxyHosts` property to a list of the hosts that you want to connect to directly.

When specifying multiple hosts, each host must be separated by a pipe (`|`). You can specify patterns using asterisks (`*`) as wildcard characters. For example:

```
NonProxyHosts=123.255.321.255|*.localhost|176.255.16.*
```

4. If the proxy server requires authentication, do the following:
 - a. Set the `ProxyUID` property to your user name for accessing the server.
 - b. Set the `ProxyPWD` property to your password for accessing the server.
 - c. To configure the driver to use the NTLM protocol, do the following:
 - i. Set the `ProxyDomain` property to the Windows domain name of the server.
 - ii. Set the `ProxyWorkstation` property to the Windows workstation name of the server.
 - d. To pre-emptively authenticate against the proxy server using basic authentication, set the `PreemptiveBasicProxyAuth` property to `1`.

If the proxy server is configured to intercept SSL-encrypted connections, then in addition to setting the connection properties described above, you must also create a keystore containing the root certificate from the proxy server.

To create a keystore for SSL interception:

1. From the proxy server, export the root certificate as a `.cer` file.
2. On your client machine, use the Java Keytool to create a keystore containing the exported root certificate:
 - a. In a command-line interface, type the following command, and then press **ENTER**:


```
[JDKInstallDir]\bin\keytool.exe -import -file  
[RootCertPath] -keystore [KeystorePath] -alias proxy
```

Where:

- *[JDKInstallDir]* is the full path to the directory where the Java Development Kit is installed.
- *[RootCertPath]* is the full path and name of the root certificate file that was exported from the proxy server.
- *[KeystorePath]* is the full path and name of the keystore that you want to create.

For example:

```
C:\Program Files\Java\jdk1.8.0\bin\keytool.exe -  
import -file  
C:\Users\jsmith\Documents\Athena\ProxyRoot.cer -  
keystore C:\Users\jsmith\AthenaKeystores -alias  
proxy
```

- b. When you are prompted to provide a password, type a password for restricting access to the keystore and then press **ENTER**.
 - c. When you are prompted to confirm your choices, type **y** and then press **ENTER**.
3. Set the following Java system properties:

```
javax.net.ssl.trustStore = [KeystorePath]
```

```
javax.net.ssl.trustStorePassword = [KeystorePassword]
```

Where:

- *[KeystorePath]* is the full path and name of the keystore containing the exported root certificate.
- *[KeystorePassword]* is the password for accessing the keystore.

Configuring Logging

To help troubleshoot issues, you can enable logging in the driver.

! Important:

Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.

In the connection URL, set the `LogLevel` key to enable logging at the desired level of detail. The following table lists the logging levels provided by the Simba Athena JDBC Driver, in order from least verbose to most verbose.

LogLevel Value	Description
0	Disable all logging.
1	Log severe error events that lead the driver to abort.
2	Log error events that might allow the driver to continue running.
3	Log events that might result in an error if action is not taken.
4	Log general information that describes the progress of the driver.
5	Log detailed information that is useful for debugging the driver.
6	Log all driver activity.

Note:

If `UseAwsLogger` is set to 1, the driver also logs information from AWS API calls.

To enable logging:

1. Set the `LogLevel` property to the desired level of information to include in log files.
2. Set the `LogPath` property to the full path to the folder where you want to save log files. To make sure that the connection URL is compatible with all

JDBC applications, escape the backslashes (\) in your file path by typing another backslash.

For example, the following connection URL enables logging level 3 and saves the log files in the `C:\temp` folder:

```
jdbc:awsathena://AwsRegion=us-east-1;UID=ABCABCABC123ABCABC45;PWD=bCD+E1f2Gxhi3J4klmN/OP5QrSTuvwXYZabcdEF;S3OutputLocation=s3://test-athena-results/;LogLevel=3;LogPath=C:\\temp
```

3. Optionally, to include information about AWS API calls in the log, set `UseAwsLogger` to 1.
4. To make sure that the new settings take effect, restart your JDBC application and reconnect to the server.

The Simba Athena JDBC Driver produces the following log files in the location specified in the `LogPath` property:

- An `AthenaJDBC_driver.log` file that logs driver activity that is not specific to a connection.
- An `AthenaJDBC_connection_[Number].log` file for each connection made to the database, where `[Number]` is a number that identifies each log file. This file logs driver activity that is specific to the connection.

If the `LogPath` value is invalid, then the driver sends the logged information to the standard output stream (`System.out`).

To disable logging:

1. Set the `LogLevel` property to 0.
2. To make sure that the new setting takes effect, restart your JDBC application and reconnect to the server.

Features

More information is provided on the following features of the Simba Athena JDBC Driver:

- [Catalog and Schema Support](#) on page 36
- [File Formats](#) on page 36
- [Fetch Size](#) on page 36
- [Data Types](#) on page 37
- [Security and Authentication](#) on page 39

Catalog and Schema Support

The Simba Athena JDBC Driver supports both catalogs and schemas to make it easy for the driver to work with various JDBC applications. Amazon Athena organizes tables into schemas/databases, and lists them under the default catalog named `AwsDataCatalog`. The data catalog can either be managed by Athena, or by AWS Glue in regions and clusters where AWS Glue has been implemented. In either case, the catalog name is `AwsDataCatalog`. The driver provides access to all of the schemas/databases that are listed under this catalog, ensuring compatibility with standard BI tools.

File Formats

The Simba Athena JDBC Driver supports all the file formats that Athena supports, which include the following:

- Avro
- Comma-Separated Values (CSV)
- JavaScript Object Notation (JSON)
- Optimized Row Columnar (ORC)
- Parquet


Fetch Size

The Simba Athena JDBC Driver supports a maximum fetch size of 1000 rows. This is consistent with the maximum fetch size that is supported by the Athena service when the result set streaming API is not used (`UseResultsetStreaming=0`).

If you use the `setFetchSize()` method from the `Statement` class to set a fetch size greater than 1000 without using the result set streaming API, the Simba Athena

JDBC Driver limits the value to 1000. When the result set streaming API is used, the driver does not impose a maximum limit on the fetch size.

If the `setFetchSize()` method is not called on the Statement object, the default fetch size is 0. In this case the fetch size is set using the `RowsToFetchPerBlock` configuration option. For more information see [RowsToFetchPerBlock](#) on page 51.

 **Note:**


While setting a large fetch size value when using the result set streaming API can give you better fetch performance, it can also result in higher memory usage. This can be mitigated if the JDBC application can retrieve the result set from the driver quickly.


Data Types

The Simba Athena JDBC Driver supports many common data formats, converting between Athena, JDBC, and Java data types.

The following table lists the supported data type mappings.

Athena Type	JDBC Type	Java Type
ARRAY	ARRAY or VARCHAR (See UseArraySupport on page 55)	java.sql.Array of strings or string
BIGINT	BIGINT	long
BINARY	VARBINARY	byte[]
BOOLEAN	BOOLEAN	boolean
CHAR	CHAR	string
DATE	DATE	java.sql.Date

 **Note:**
Not supported for Parquet files.

Athena Type	JDBC Type	Java Type
DECIMAL (p, s)	DECIMAL	java.math.BigDecimal
DOUBLE	DOUBLE	double
FLOAT	REAL	float
INTEGER <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> Note: Although Athena reports integer data as type INT, the driver reports integer data as type INTEGER to ensure compatibility with standard BI tools. For more information, see Integer Support on page 38.</p> </div>	INTEGER	int
MAP	VARCHAR	String
SMALLINT	SMALLINT	short
STRING	VARCHAR	String
STRUCT	VARCHAR	String
TIMESTAMP	TIMESTAMP	java.sql.Timestamp
TINYINT	TINYINT	byte
VARCHAR	VARCHAR	String

Integer Support

Athena combines two different implementations of the integer data type:

- In Data Definition Language (DDL) queries, Athena uses the INT data type from Apache Hive.
- In all other queries, Athena uses the INTEGER data type from Presto.

To support the CAST queries that are used in many BI tools, the driver reports integer data as type INTEGER even though Athena reports the data as type INT.

Be aware that, when executing DDL queries, you must specify integer data using INT as the data type.

**Note:**

Athena supports some but not all DDL statements. For a list of the supported DDL statements, see "SQL and HiveQL Reference" in the *Amazon Athena API Reference*: <http://docs.aws.amazon.com/athena/latest/ug/language-reference.html>.

Integration with AWS Glue

Support for AWS Glue is integrated into Simba Athena JDBC Driver. AWS Glue is a fully managed ETL (extract, transform, and load) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores.

For optimal detection of AWS Glue, the IAM user for the driver requires permissions for the `glue:GetCatalogImportStatus` API in its policy. The default AWS Managed Athena policy, `AmazonAthenaFullAccess`, does not grant access to this API by default. Refer to your Amazon Web Services documentation for information on how to grant API access in the policy settings. Without the proper permission to this API, the driver falls back to the legacy detection logic at connection time, which may impact driver performance.

For a full description of AWS Glue, see

<https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html>.

For more information about AWS Glue integration, see

https://docs.aws.amazon.com/en_us/athena/latest/ug/glue-athena.html.


Security and Authentication

To protect data from unauthorized access, Athena requires all connections to be authenticated using an access key and a secret key, and uses the SSL protocol that is implemented in Amazon Web Services. The Simba Athena JDBC Driver protects your

data by providing support for these authentication protocols and further obscuring data from unwanted access by providing encryption options for your query results.

The driver provides mechanisms that enable you to authenticate your connection using either an AWS access key and secret key, or a class that implements the `AWSCredentialsProvider` interface. For detailed configuration instructions, see [Configuring Authentication](#) on page 27.

Additionally, the driver automatically applies SSL encryption to all connections. SSL encryption protects data and credentials when they are transferred over the network, and provides stronger security than authentication alone.

 **Note:**

In this documentation, "SSL" indicates both TLS (Transport Layer Security) and SSL (Secure Sockets Layer). The driver supports industry-standard versions of TLS/SSL.

The SSL version that the driver supports depends on the JVM version that you are using. For information about the SSL versions that are supported by each version of Java, see "Diagnosing TLS, SSL, and HTTPS" on the Java Platform Group Product Management Blog: https://blogs.oracle.com/java-platform-group/entry/diagnosing_tls_ssl_and_https.

 **Note:**


The SSL version used for the connection is the highest version that is supported by both the driver and the server, which is determined at connection time.

For query results, the Simba Athena JDBC Driver supports all the encryption options that Athena supports. For detailed information about the supported encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*: <http://docs.aws.amazon.com/athena/latest/ug/encryption.html>. For information about configuring encryption in the driver, see [Configuring Query Result Encryption](#) on page 31.

Driver Configuration Options

Driver Configuration Options lists and describes the properties that you can use to configure the behavior of the Simba Athena JDBC Driver.

You can set configuration properties using the connection URL. For more information, see [Building the Connection URL](#) on page 13.

 **Note:**

Property names and values are case-sensitive.

AwsCredentialsProviderArguments

Default Value	Data Type	Required
None	String	Yes, if <code>UID</code> and <code>PWD</code> are not provided, and if <code>AwsCredentialsProviderClass</code> does not have a default constructor.

Description

A comma-separated list of String arguments for the constructor of the `AwsCredentialsProviderClass`.

Be aware of the following restrictions:

- The driver only supports String arguments for the constructor parameters.
- Multiple arguments must be separated by a comma (,).
- Surrounding spaces are not included in the parsed arguments.
- To escape a single character, use a backslash (\) before that character. To indicate a backslash in an argument, use two backslashes (\ \).
- To escape all commas in an argument, enclose the argument in quotation marks ("). To indicate a quotation mark in a quoted argument, use a backslash (\) before that quotation mark.

For detailed instructions on configuring authentication using the `AWSCredentialsProvider` interface, see [Using the AWSCredentialsProvider Interface](#) on page 27.

This can also be configured using the alias `aws_credentials_provider_arguments`.

AwsCredentialsProviderClass

Default Value	Data Type	Required
None	String	Yes, if <code>UID</code> and <code>PWD</code> are not provided.

Description

The fully qualified name of a class that implements the `AWSCredentialsProvider` interface.

! Important:

- If you use a class implementation from the AWS SDK, use the shaded package name for amazonaws that is included inside the driver jar. This is `com.simba.athena.amazonaws`.
- If you use a custom class implementation, include that implementation in your class path. In addition, import the amazonaws classes using the shaded package name, `com.simba.athena.amazonaws`.

For detailed instructions on configuring authentication using the `AWSCredentialsProvider` interface, see [Using the AWSCredentialsProvider Interface](#) on page 27.

This can also be configured using the alias `aws_credentials_provider_class`.

AwsRegion

Default Value	Data Type	Required
None	String	Yes

Description

The AWS region of the Athena and AWS Glue instance that you want to connect to.

The region can also be taken from the endpoint provided in the connection string `jdbc:awsathena://athena.[Region].amazonaws.com:443;`. The region

will be parsed out of this endpoint and used for connecting to Athena and AWS Glue services. If both are present in the connection string the `AWSRegion` takes precedence.

For a list of valid regions, see the "Athena" section in the *AWS Regions and Endpoints* documentation: <http://docs.aws.amazon.com/general/latest/gr/rande.html#athena>.

BinaryColumnLength

Default Value	Data Type	Required
32767	Integer	No

Description

The maximum data length for BINARY columns.

ComplexTypeColumnLength

Default Value	Data Type	Required
65535	Integer	No

Description

The maximum data length for ARRAY, MAP, and STRUCT columns.

ConnectionTest

Default Value	Data Type	Required
1	Integer	No

Description

This property determines whether the driver should verify connection by sending a simple "SELECT 1" query during establishing a connection with Athena.

1 : The driver verifies connection by sending a simple "SELECT 1" query to Athena.

0 : The driver does not send any query to Athena to verify the connection.

! Important:

Setting the value to 0 means that driver will not verify the connection. The connection string may contain unverified configuration values, such as incorrect authentication information, which will not be discovered at connection. This can result in errors when the application attempts to execute a query or any other JDBC API calls using the driver.

ConnectTimeout

Default Value	Data Type	Required
10	Integer	No

Description

The amount of time, in seconds, that the driver waits when establishing a connection before timing out the connection.

A value of 0 indicates that the driver never times out the connection.

! Important:

Setting this property to 0 is not recommended.

This can also be configured using the alias `connection_timeout`. If this is used then the amount of time is measured in milliseconds.

LogLevel

Default Value	Data Type	Required
0	Integer	No

Description

Use this property to enable or disable logging in the driver and to specify the amount of detail included in log files.

! Important:

Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.

Set the property to one of the following numbers:

- 0: Disable all logging.
- 1: Enable logging on the FATAL level, which logs very severe error events that will lead the driver to abort.
- 2: Enable logging on the ERROR level, which logs error events that might still allow the driver to continue running.
- 3: Enable logging on the WARNING level, which logs events that might result in an error if action is not taken.
- 4: Enable logging on the INFO level, which logs general information that describes the progress of the driver.
- 5: Enable logging on the DEBUG level, which logs detailed information that is useful for debugging the driver.
- 6: Enable logging on the TRACE level, which logs all driver activity.

**Note:**

If `UseAwsLogger` is set to 1, the driver also logs information from AWS API calls. See [OUseAwsLogger](#) on page 55.

When logging is enabled, the driver produces the following log files in the location specified in the `LogPath` property:

- An `AthenaJDBC_driver.log` file that logs driver activity that is not specific to a connection.
- An `AthenaJDBC_connection_[Number].log` file for each connection made to the database, where `[Number]` is a number that distinguishes each log file from the others. This file logs driver activity that is specific to the connection.

If the `LogPath` value is invalid, then the driver sends the logged information to the standard output stream (`System.out`).

LogPath

Default Value	Data Type	Required
The current working directory.	String	No

Description

The full path to the folder where the driver saves log files when logging is enabled.

MaxCatalogNameLength

Default Value	Data Type	Required
0	Integer	No

Description

The maximum number of characters that catalog names can contain.

To indicate that there is no maximum length or that the length is unknown, set this option to 0.

MaxColumnNameLength

Default Value	Data Type	Required
0	Integer	No

Description

The maximum number of characters that column names can contain.

To indicate that there is no maximum length or that the length is unknown, set this option to 0.

MaxErrorRetry

Default Value	Data Type	Required
10	Integer	No

Description

The maximum number of times that the driver resubmits a failed request that can be retried, such as a 5xx error from the Athena server.

**Note:**

Do not specify a negative value for this setting.

This can also be configured using the alias `max_error_retries`.

MaxSchemaNameLength

Default Value	Data Type	Required
256	Integer	No

Description

The maximum number of characters that schema names can contain.

To indicate that there is no maximum length or that the length is unknown, set this option to 0.

MaxTableNameLength

Default Value	Data Type	Required
0	Integer	No

Description

The maximum number of characters that table names can contain.

To indicate that there is no maximum length or that the length is unknown, set this option to 0.

MetadataRetrievalMethod

Default Value	Data Type	Required
Auto	String	No

Description

This property determines how the metadata would be retrieved from Athena for different JDBC API calls like `getTables`, `getColumns`. Following are the valid values:

- `Auto`: During connection time driver will automatically determine whether to use AWS Glue or Query to get metadata for the specified Athena region. If AWS Glue is supported in the region and Athena has been upgraded to use AWS Glue, driver will use AWS Glue to get the metadata. If AWS Glue is not supported in the region or Athena hasn't been upgraded to use AWS Glue, driver will query Athena to get the metadata.
- `Glue`: Driver will use AWS Glue to get the metadata regardless of whether AWS Glue is supported or used in the region.
- `Query`: Driver will use Query to get the metadata regardless of whether AWS Glue is supported or used in that region.

! Important:

Changing the default value for this configuration option may lead to unwanted behavior. For example, the driver may attempt to use AWS Glue in a region where AWS Glue is not supported or used.

NonProxyHosts

Default Value	Data Type	Required
None	String	No

Description

A list of hosts that the driver can access without connecting through the proxy server, when a proxy connection is enabled.

When specifying multiple hosts, each host must be separated by a vertical bar (`|`). You can specify patterns using asterisks (`*`) as wildcard characters. For example:

```
NonProxyHosts=123.255.321.255|*.localhost|176.255.16.*
```


PWD

Default Value	Data Type	Required
None	String	Yes, if <code>AwsCredentialsProviderClass</code> is not provided.

Description

The secret key provided by your AWS account.

This can also be configured using the alias `password`.

PreemptiveBasicProxyAuth

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the driver pre-emptively authenticates against the proxy server using basic authentication, when a proxy connection is enabled.

- 1: The driver pre-emptively authenticates the connection using basic authentication.
- 0: The driver does not pre-emptively authenticate the connection using basic authentication.

ProxyDomain

Default Value	Data Type	Required
None	String	No

Description

The Windows domain name of the server that you want to authenticate through, when authenticating a proxy connection using the NTLM protocol.

ProxyHost

Default Value	Data Type	Required
None	String	No

Description

The IP address or host name of your proxy server.

ProxyPort

Default Value	Data Type	Required
None	Integer	No

Description

The listening port of your proxy server.

ProxyPWD

Default Value	Data Type	Required
None	String	Yes, if connecting through a proxy server that requires authentication.

Description

The password that you use to access the proxy server.

ProxyUID

Default Value	Data Type	Required
None	String	Yes, if connecting through a proxy server that requires authentication.

Description

The user name that you use to access the proxy server.

ProxyWorkstation

Default Value	Data Type	Required
None	String	No

Description

The Windows workstation name of the server that you want to authenticate through, when authenticating a proxy connection using the NTLM protocol.

RowsToFetchPerBlock

Default Value	Data Type	Required
10000 for result set streaming, 1000 for pagination	Integer	No

Description

The maximum number of rows to fetch per stream if using the result set streaming API. The maximum number of rows to fetch per page if using pagination.

Note:

While setting this option with a large value when using the result set streaming API can give you better fetch performance, it can also result in higher memory usage. This can be mitigated if the JDBC application can retrieve the result set from the driver quickly.

See [UseResultSetStreaming](#) on page 56 for details.

S3OutputEncKMSKey

Default Value	Data Type	Required
None	String	Yes, if using SSE_KMS or CSE_KMS encryption.

Description

The KMS key ARN or ID to use when encrypting query results using SSE_KMS or CSE_KMS encryption.

For detailed information about the supported encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*:

<http://docs.aws.amazon.com/athena/latest/ug/encryption.html>.

This can also be configured using the alias `query_results_aws_kms_key`.

S3OutputEncOption

Default Value	Data Type	Required
None	String	No

Description

The encryption protocol that the driver uses to encrypt your query results before storing them on Amazon S3.

- `SSE_S3`: The driver uses server-side encryption with an Amazon S3-managed key.

- `SSE_KMS`: The driver uses server-side encryption with an AWS KMS-managed key.
- `CSE_KMS`: The driver uses client-side encryption with an AWS KMS-managed key.

**Note:**

If this value is not set, the driver does not encrypt the query results before storing them on Amazon S3.

For detailed information about these encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*:

<http://docs.aws.amazon.com/athena/latest/ug/encryption.html>.

This can also be configured using the alias `query_results_encryption_option`.

S3OutputLocation

Default Value	Data Type	Required
None	String	Yes

Description

The path of the Amazon S3 location where you want to store query results, prefixed by `s3://`.

For example, to store Athena query results in a folder named "test-folder-1" inside an S3 bucket named "query-results-bucket", you would set this property to `s3://query-results-bucket/test-folder-1`.

This can also be configured using the alias `s3_staging_dir`.

Schema

Default Value	Data Type	Required
default	String	No

Description

The name of the database schema to use when a schema is not explicitly specified in a query. You can still issue queries on other schemas by explicitly specifying the schema in the query.

SocketTimeout

Default Value	Data Type	Required
50	Integer	No

Description

The amount of time, in seconds, that the driver waits for data to be transferred over an established, open connection before timing out the connection.

A value of 0 indicates that the driver never times out the connection.

! Important:

Setting this property to 0 is not recommended.

This can also be configured using the alias `socket_timeout`. If this option is used, the time is measured in milliseconds.

StringColumnLength

Default Value	Data Type	Required
255	Integer	No

Description

The maximum data length for STRING columns.

UseArraySupport

Default Value	Data Type	Required
1	Integer	No

Description

This property specifies whether the driver supports getting the ResultSet data as an array.

- 1: The driver returns ResultSet data as an array.
- 0: The driver returns ResultSet as VARCHAR.

The driver makes the following assumptions when returning the ResultSet as an array:

- The array columns are always of type `Array<String>`.
- The array column data in the result set start and end with bracket characters ([and]), and the array elements are delimited by commas (,). This means that multidimensional arrays or array elements that contain commas in their values are not parsed correctly.

OUseAwsLogger

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the driver records the log output from any AWS API calls. For information about logging, see [Configuring Logging](#) on page 34.

- 1: If logging is enabled, the driver records the log outputs from any AWS API calls in the driver log file.
- 0: The driver does not log AWS API calls.

UseResultsetStreaming

Default Value	Data Type	Required
1	Integer	No

Description

This property specifies whether the driver uses the AWS result set streaming API for result set fetching.

- 1: The driver uses the result set streaming API.
- 0: The driver uses pagination logic for result set fetching.

UID

Default Value	Data Type	Required
None	String	Yes, if <code>AwsCredentialsProviderClass</code> is not provided.

Description

The access key provided by your AWS account.

This can also be configured using the alias `user`.

Appendix: Migrating to Later Driver Versions

This appendix contains information to help you successfully migrate from the 1.x and 2.x versions of the driver to the latest versions of the driver.

The following sections list differences between drivers that may disrupt workflows when you migrate, and provides recommendations on how to recreate those workflows for a successful migration.

Upgrading From 1.x to 2.0.x

JDBC Driver Class Name

The drivers use different class names.

Version 1.x	Version 2.x
<code>com.amazonaws.athena.jdbc.AthenaDriver</code>	<code>com.simba.athena.jdbc.Driver</code>

If you are using the following line in your code to explicitly load the driver class in your source code:

```
Class.forName("com.amazonaws.athena.jdbc.AthenaDriver");
```

then you will need to change it to:

```
Class.forName("com.simba.athena.jdbc.Driver");
```

Connection URL

Specifying the Host and Port

The 2.x version provides an alternative way to specify the AWS region.

1.x Version	2.x Version
<pre>jdbc:awsathena://athena. {REGION}.amazonaws.com:443</pre> <p>Where {REGION} is a region identifier, such as us-west-2</p>	<pre>jdbc:awsathena://athena. {REGION}.amazonaws.com:443</pre> <p>Or</p> <pre>jdbc:awsathena://AwsRegion= {REGION}</pre> <p>Where {REGION} is a region identifier, such as us-west-2. If {REGION} is specified using both endpoint URL and AwsRegion, the value specified in AwsRegion takes precedence.</p>

Changes are not required in this case, but be aware the 2.x version provides an alternative way to specify the AWS region in the connection URL.

Connection String Attributes Separator

The drivers use different attribute separators in their connection URLs.

1.x Version	2.x Version
& and ?	;

The following is an example connection URL using the 1.x version syntax:

```
jdbc:awsathena://athena.us-west-1.amazonaws.com:443?s3_
staging_dir=s3://query-resultsbucket/folder/&query_results_
encryption_option=SSE_S3
```

The following shows the equivalent URL constructed using the 2.x version syntax:

```
jdbc:awsathena://athena.us-west-1.amazonaws.com:443;s3_
staging_dir=s3://query-resultsbucket/folder/;query_results_
encryption_option=SSE_S3
```

Driver Configuration Options

There are some differences in the supported connection properties for the drivers.

Version 1.x Option	Version 2.x Option	Possible Values
log_path	LogPath	No difference.

Version 1.x Option	Version 2.x Option	Possible Values	
log_level	LogLevel	1.x	2.x
		OFF	0
		FATAL	1
		ERROR	2
		WARNING	3
		INFO	4
		DEBUG	5
TRACE	6		
retry_base_delay	Not configurable.		
retry_max_backoff_time	Not configurable.		

The following is an example connection URL for enabling logging using the syntax for version 1.x:

```
jdbc:awsathena://athena.us-west-1.amazonaws.com:443?s3_staging_dir=s3://query-resultsbucket/folder/&log_level=TRACE&log_path=/tmp
```

The following is the equivalent connection URL using the syntax for version 2.x:

```
jdbc:awsathena://athena.us-west-1.amazonaws.com:443;s3_staging_dir=s3://query-resultsbucket/folder/;LogLevel=6;LogPath=/tmp
```

ResultSetMetaData Differences for API Calls

The drivers return different metadata for the following API calls.

getCatalogs

Column Name	Version 1.x		Version 2.x	
TABLE_CAT	Metadata	Value	Metadata	Value
	Type Name	varchar	Type Name	VARCHAR
	Type ID	-16	Type ID	12
	Display Size	1073741824	Display Size	128
	Precision	1073741824	Precision	128
	Scale	0	Scale	0

getColumnns

Column Name	Version 1.x		Version 2.x	
TABLE_CAT TABLE_SCHEM TABLE_NAME COLUMN_NAME TYPE_NAME IS_AUTOINCREMENT IS_GENERATEDCOLUMN	Metadata	Value	Metadata	Value
	Type Name	varchar	Type Name	VARCHAR
	Type ID	-16	Type ID	12
	Display Size	1073741824	Display Size	128
	Precision	1073741824	Precision	128
	Scale	0	Scale	0

Column Name	Version 1.x		Version 2.x	
REMARKS COLUMN_DEF IS_NULLABLE SCOPE_CATALOG SCOPE_SCHEMA SCOPE_TABLE	Metadata	Value	Metadata	Value
	Type Name	varchar	Type Name	VARCHAR
	Type ID	-16	Type ID	12
	Display Size	1073741824	Display Size	254
	Precision	1073741824	Precision	254
	Scale	0	Scale	0
SOURCE_DATA_TYPE	Metadata	Value	Metadata	Value
	Type Name	smallint	Type Name	INGEGER
	Type ID	5	Type ID	4
	Display Size	6	Display Size	11
	Precision	5	Precision	10
	Scale	0	Scale	0

getSchemas

Column Name	Version 1.x		Version 2.x	
TABLE_SCHEM TABLE_CATALOG	Metadata	Value	Metadata	Value
	Type Name	varchar	Type Name	VARCHAR
	Type ID	-16	Type ID	12
	Display Size	1073741824	Display Size	128
	Precision	1073741824	Precision	128
	Scale	0	Scale	0

getTables

Column Name	Version 1.x		Version 2.x	
TABLE_CAT TABLE_SCHEM TABLE_NAME TABLE_TYPE TYPE_CAT TYPE_SCHEM TYPE_NAME SELF_REFERENCING_ COL_NAME REF_GENERATION	Metadata	Value	Metadata	Value
	Type Name	varchar	Type Name	VARCHAR
	Type ID	-16	Type ID	12
	Display Size	1073741824	Display Size	128
	Precision	1073741824	Precision	128
	Scale	0	Scale	0
	REMARKS	Metadata	Value	Metadata
Type Name		varchar	Type Name	VARCHAR
Type ID		-16	Type ID	12
Display Size		1073741824	Display Size	254
Precision		1073741824	Precision	254
Scale		0	Scale	0

Data Type for TIME Literal in Query Result

For a query such as `SELECT TIME '12:00:00'`, the drivers use different data types in the query result set for the TIME literal column.

Version 1.x	Version 2.x
TIME	VARCHAR

Upgrading from 2.0.2 to 2.0.5 or later

Result Set Streaming Support

Starting with version 2.0.5, the driver uses the result set streaming API to improve the performance in fetching query results. To take advantage of this feature you must include and allow the `athena:GetQueryResultsStream` action in your IAM policy statement. For details on managing Athena IAM policies, see <https://docs.aws.amazon.com/athena/latest/ug/access.html>.

Third-Party Trademarks

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Amazon Athena, Amazon S3, Amazon Simple Storage Service, Amazon Web Services, AWS, AWS Glue, and Amazon are trademarks or registered trademarks of Amazon Web Services, Inc. or its subsidiaries in Canada, United States and/or other countries.

All other trademarks are trademarks of their respective owners.

Third-Party Licenses

The licenses for the third-party libraries that are included in this product are listed below.

Apache License, Version 2.0

The following notice is included in compliance with the Apache License, Version 2.0 and is applicable to all software licensed under the Apache License, Version 2.0.

Apache License

Version 2.0, January 2004

<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is

included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
 - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
 - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
 - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
 - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required

for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product includes software that is licensed under the Apache License, Version 2.0 (listed below):

AWS SDK for Java Core

Copyright © 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Apache Commons Codec

Copyright © 2002-2014 The Apache Software Foundation

Apache Commons CSV

Copyright © 2017 The Apache Software Foundation

Apache Commons Logging

Copyright © 2001-2014 The Apache Software Foundation

Apache HttpComponents

Copyright © 2005-2015 The Apache Software Foundation

Apache Log4j

Copyright © 1999-2014 The Apache Software Foundation

This product includes software developed by The Apache Software Foundation (<http://www.apache.org/>).

ResolverUtil.java

Copyright 2005-2006 Tim Fennell

Dumbster SMTP test server

Copyright 2004 Jason Paul Kitchen

TypeUtil.java

Copyright 2002-2012 Ramnivas Laddad, Juergen Hoeller, Chris Beams

Jackson

Copyright © 2009-2011 FasterXML, LLC

Joda-Time

Copyright © 2005–2015 Joda.org. All rights reserved.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.