



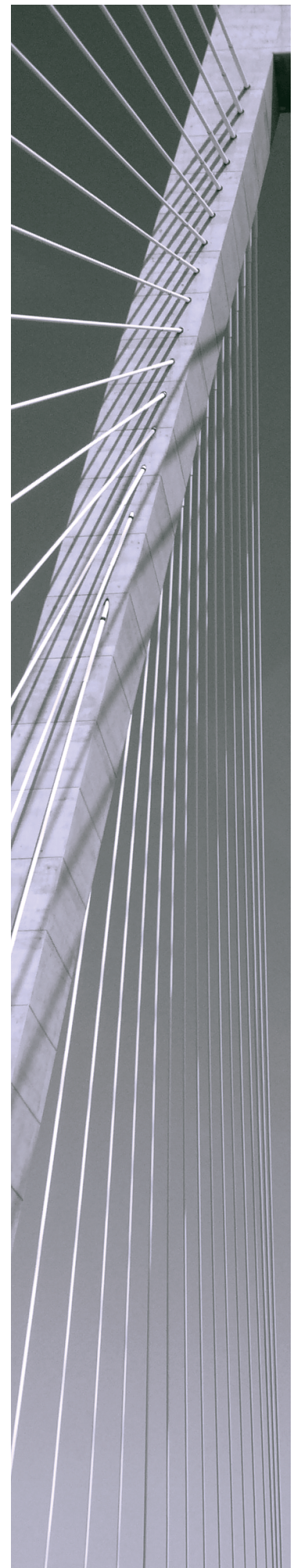
**SIMBA**<sup>®</sup>  
**BY MAGNITUDE**

# Simba Athena JDBC Driver with SQL Connector

## Installation and Configuration Guide

Simba Technologies Inc.

Version 2.0.9  
August 30, 2019



**Copyright © 2020 Magnitude Software, Inc. All rights reserved.**

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from Magnitude.

The information in this document is subject to change without notice. Magnitude strives to keep this information accurate but does not warrant that this document is error-free.

Any Magnitude product described herein is licensed exclusively subject to the conditions set forth in your Magnitude license agreement.

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

All other company and product names mentioned herein are used for identification purposes only and may be trademarks or registered trademarks of their respective owners.

Information about the third-party products is contained in a third-party-licenses.txt file that is packaged with the software.

**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](http://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 .....	8
About Amazon Athena .....	8
About the Driver .....	8
System Requirements .....	10
Simba Athena JDBC Driver Files .....	11
Installing and Using the Simba Athena JDBC Driver .....	12
Referencing the JDBC Driver Libraries .....	12
Registering the Driver Class .....	13
Building the Connection URL .....	17
Example: Using Workgroups .....	20
Example: Using the Driver in SQL Workbench .....	21
Examples: Using the Driver in a Java Application .....	27
Configuring Authentication .....	33
Using IAM Credentials .....	33
Using the AWSCredentialsProvider Interface .....	33
Using the Active Directory Federation Services (AD FS) Credentials Provider .....	38
Using the Okta Credentials Provider .....	44
Using a Custom Credentials Provider for an Identity Provider .....	49
Using the Post-SAML Workflow Hook .....	57
Configuring Query Result Encryption .....	63
Configuring Proxy Connections .....	64
Configuring Logging .....	66
Configuring Connection Timeouts and Retries .....	68
Features .....	69
Catalog and Schema Support .....	69
File Formats .....	69
Fetch Size .....	69
Query Execution Polling .....	70
Data Types .....	71
Query ID Retrieval .....	73
Integration with AWS Glue .....	74

Security and Authentication .....	74
Driver Configuration Options .....	76
App_ID .....	76
App_Name .....	76
AwsCredentialsProviderArguments .....	77
AwsCredentialsProviderClass .....	78
AwsRegion .....	79
BinaryColumnLength .....	79
ComplexTypeColumnLength .....	80
ConnectionTest .....	80
ConnectTimeout .....	80
EndpointOverride .....	81
IdP_Host .....	82
IdP_Port .....	82
LogLevel .....	82
LogPath .....	84
MaxCatalogNameLength .....	84
MaxColumnNameLength .....	84
MaxErrorRetry .....	85
MaxQueryExecutionPollingInterval .....	85
MaxSchemaNameLength .....	86
MaxStreamErrorRetry .....	86
MaxTableNameLength .....	86
MinQueryExecutionPollingInterval .....	87
NonProxyHosts .....	87
Password .....	88
PreemptiveBasicProxyAuth .....	88
preferred_role .....	89
Profile .....	89
ProxyDomain .....	90
ProxyHost .....	90
ProxyPort .....	90
ProxyPWD .....	91
ProxyUID .....	91
ProxyWorkstation .....	91
QueryExecutionPollingIntervalMultiplier .....	92
RowsToFetchPerBlock .....	92

# Installation and Configuration Guide

S3OutputEncKMSKey .....	93
S3OutputEncOption .....	93
S3OutputLocation .....	94
Schema .....	95
SocketTimeout .....	95
SSL_Insecure .....	96
StreamingEndpointOverride .....	96
StringColumnLength .....	96
UseArraySupport .....	97
UseAwsLogger .....	97
User .....	98
UseResultsetStreaming .....	98
Workgroup .....	99
Appendix: Migrating to Later Driver Versions .....	100
Upgrading From 1.x to 2.0.x .....	100
Upgrading from 2.0.2 to 2.0.5 or later .....	106
Third-Party Trademarks .....	107

## 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.

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) 7.0 or 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 12)
- The `Driver` or `DataSource` class (see [Registering the Driver Class](#) on page 13)
- The connection URL for the driver (see [Building the Connection URL](#) on page 17)

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 21.
- 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 27.

## 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 classes are used to connect the Simba Athena JDBC Driver to the Athena service:

- The `Driver` classes extend `java.sql.Driver`.
- 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` class to establish a connection for JDBC:

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

```

When using the `DataSource` class to connect to Athena, you can call the following methods to configure `DataSource` properties:

Method	Description
<code>setCustomProperty(String key, String value)</code>	<p>Set a property that is not otherwise supported by one of the set/get methods listed in this table.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li>• <code>key</code>: The key name of the property.</li> <li>• <code>value</code>: The value that you want to specify for the property.</li> </ul> <p>After setting a custom property, you can access it by calling the <code>getCustomProperty()</code> method.</p>

Method	Description
<code>setLogDirectory(String logDirectory)</code>	<p>Set the directory that the driver uses for logging.</p> <p>Parameters:</p> <ul style="list-style-type: none"><li>• <code>logDirectory</code>: The directory where you want log files to be saved.</li></ul> <p>This property is equivalent to the connection-level property <a href="#">LogPath</a> on page 84, which can be configured in the connection URL.</p>
<code>setLoginTimeout(int seconds)</code>	<p>Set the amount of time that the driver waits before timing out during login attempts.</p> <p>Parameters:</p> <ul style="list-style-type: none"><li>• <code>seconds</code>: The number of seconds to wait before timing out a login attempt.</li></ul> <p>This property is equivalent to the connection-level property <a href="#">ConnectTimeout</a> on page 80, which can be configured in the connection URL.</p>

Method	Description
<code>setLogLevel(String level)</code>	<p>Set the logging level, which specifies the level of detail that the driver includes in log files.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li><code>level</code>: The logging level to use, indicated by a number ranging from 0 to 6 (inclusive). 0 disables logging, while 6 is the most detailed logging level.</li> </ul> <p>This property is equivalent to the connection-level property <a href="#">LogLevel</a> on page 82, which can be configured in the connection URL.</p>
<code>setLogWriter(PrintWriter writer)</code>	<p>Set the writer that the driver uses for logging.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li><code>writer</code>: The <code>java.io.PrintWriter</code> object to use for writing the log.</li> </ul>
<code>setPassword(String pass)</code>	<p>Set the password to use for authenticating the connection. Connections to Athena require a secret key instead of an account password.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li><code>pass</code>: The secret key provided by your AWS account.</li> </ul> <p>This property is equivalent to the connection-level property <a href="#">Password</a> on page 88, which can be configured in the connection URL.</p>

Method	Description
<code>setURL(String url)</code>	<p>Set the connection URL to use for connecting to Athena.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li><code>url</code>: The connection URL. For more information, see <a href="#">Building the Connection URL</a> on page 17.</li> </ul>
<code>setUserID(String id)</code>	<p>Set the user ID to use for authenticating the connection. Connections to Athena require an access key instead of the ID of a user account.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li><code>id</code>: The access key provided by your AWS account.</li> </ul> <p>This property is equivalent to the connection-level property <a href="#">User</a> on page 98, which can be configured in the connection URL.</p>

## Building the Connection URL

Use the connection URL to supply connection information to the data store that you are accessing. The following is the format of the connection URL for the Simba Athena JDBC Driver:

```
jdbc:awsathena://User=[AccessKey];Password=[SecretKey];S3OutputLocation=[Output];[Property1]=[Value1];[Property2]=[Value2];...
```

The following variables are used in the example connection strings:

- `[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 76.

**! Important:**

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

## Standard Connection String

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

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

For example, to connect to Athena and authenticate the connection using a user name and password, you would use the following connection URL:

```
jdbc:awsathena://AwsRegion=us-west-1;User=FJ4EK9SE02Kfk;Password=P9PcyQ7prK2LwUMZMpFQ4R+5VE;S3OutputLocation=s3://working/dir;
```

## Using an Endpoint URL

The driver supports two types of endpoint URLs: a default endpoint format, and a VPC endpoint format.

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

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

For example, to connect to Athena using a standard endpoint, you would use the following connection URL:

```
jdbc:awsathena://athena.us-east-1.com:443;User=FJ4EK9SE02Kfk;Password=P9PcyQ7prK2LwUMZMpFQ4R+5VE;S3OutputLocation=s3://working/dir;
```

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

```
jdbc:awsathena://[vpce-specific-url].athena.
```

```
[Region].amazonaws.com:443;User=[AccessKey];Password=[SecretKey];S3OutputLocation=[Output];[Property1]=[Value1];[Property2]=[Value2];...
```

For example, to connect to Athena using a VPC endpoint, you would use the following connection URL:

```
jdbc:awsathena://vpce-73hddf25rus619fh62-8wnf52jd-eu-west-2a.athena.eu-west-2.vpce.amazonaws.com:443;User=FJ4EK9SE02Kfk;Password=P9PcyQ7prK2LwUMZMpFQ4R+5VE;S3OutputLocation=s3://working/dir;
```

 **Note:**

If the endpoint is specified in the URL, as shown in the examples above, then the `AwsRegion` property should not be used.

## Using Endpoint Overrides

The Athena endpoint can also be specified in the `EndpointOverride` connection property.

The following are two possible formats for connection URLs where the endpoint is specified in the `EndpointOverride` connection property. The first uses a default endpoint, while the second uses a VPC endpoint.

Default endpoint:

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

VPC endpoint:

```
jdbc:awsathena://EndpointOverride=[vpce-specific-url].athena.[Region].vpce.amazonaws.com:443;User=[AccessKey];Password=[SecretKey];S3OutputLocation=[Output];[Property1]=[Value1];[Property2]=[Value2];...
```

 **Note:**

If endpoints are specified in both the URL and the `EndpointOverride` connection property, the endpoint in the connection property takes precedence.

Additionally, if you are using the Athena streaming service, but are not using the default endpoint for this service, you can use the `StreamingEndpointOverride` connection property to specify the necessary endpoint.

The following is the format of a connection URL that uses a VPC endpoint and specifies the `StreamingEndpointOverride` property:

```
jdbc:awsathena://[vpce-specific-url].athena.  
[Region].vpce.amazonaws.com:443;StreamingEndpointOverride=  
[StreamingEndpoint]User=[AccessKey];Password=  
[SecretKey];S3OutputLocation=[Output];[Property1]=[Value1];  
[Property2]=[Value2];...
```

 **Note:**

Typically, the `StreamingEndpointOverride` property is not required. If this property is not specified, the driver tries to access the Athena streaming service using port 444 of the specified Athena instance.

## Example: Using Workgroups

A workgroup is an Athena feature that enables you to control the data access and costs associated with running queries. For more information, see "Using Workgroups to Control Query Access and Costs" in the *Amazon Athena User Guide*:

<https://docs.aws.amazon.com/athena/latest/ug/manage-queries-control-costs-with-workgroups.html>.

To use a workgroup when connecting to Athena through the Simba Athena JDBC Driver, write a connection URL that sets the `Workgroup` property to the name of your workgroup. For example, to use a workgroup named `SimbaAdmins`:

```
jdbc:awsathena://AwsRegion=us-west-  
1;User=FJ4EK9SE02Kfk;Password=P9PcyQ7prK2LwUMZMpFQ4R+5VE;S3O  
utputLocation=s3://working/dir;Workgroup=SimbaAdmins;
```

Specifying a workgroup does not change the way that you run SQL statements or make JDBC API calls. The driver passes the workgroup name to Athena, and all workgroup handling takes place in the Athena service instead of in the driver.

## 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.

The following topics are discussed here:

- [Configuring SQL Workbench to Use the Driver](#) on page 21
- [Querying Data with SQL Workbench](#) on page 24
- [Exploring Data with SQL Workbench](#) on page 25

### 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

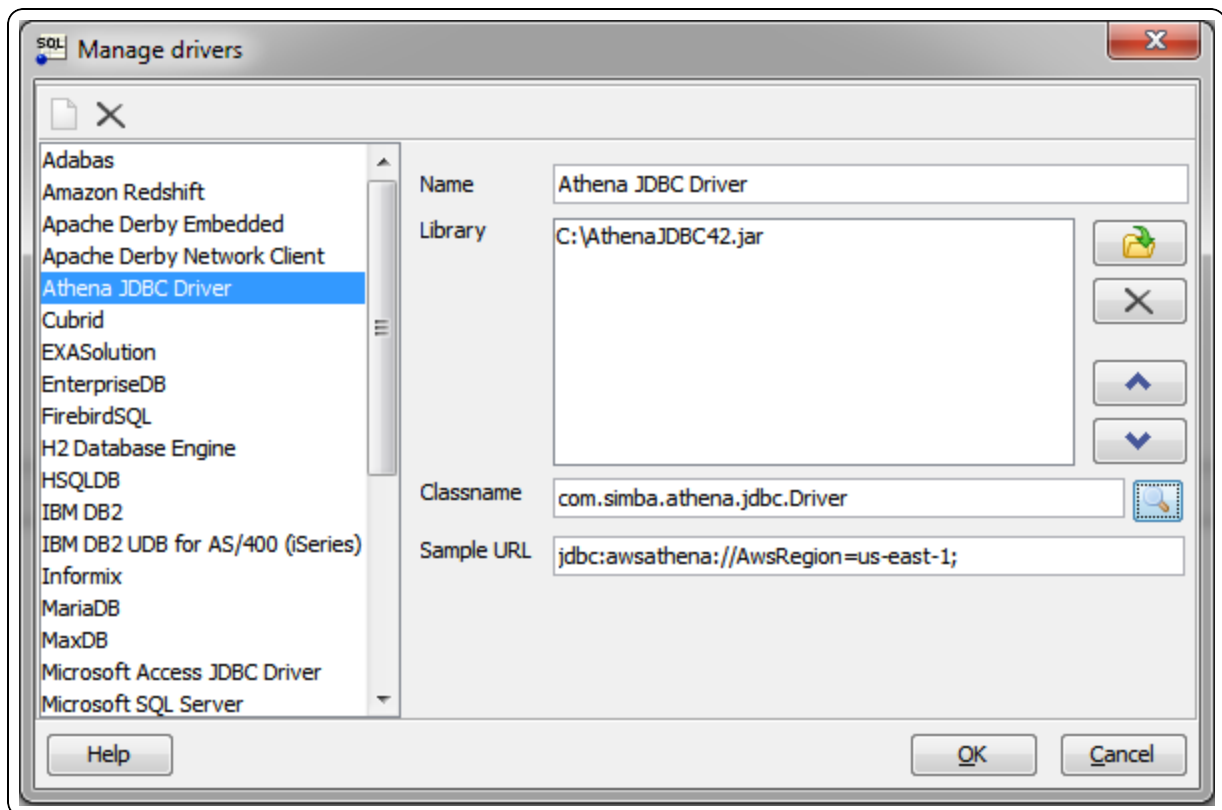
Add the Simba Athena JDBC Driver to the list of drivers in SQL Workbench, and then create a connection profile that contains the necessary connection information.

**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
<b>Name</b>	A name that you want to use to identify the Simba Athena JDBC Driver in SQL Workbench.  For example, <b>Athena JDBC Driver</b> .

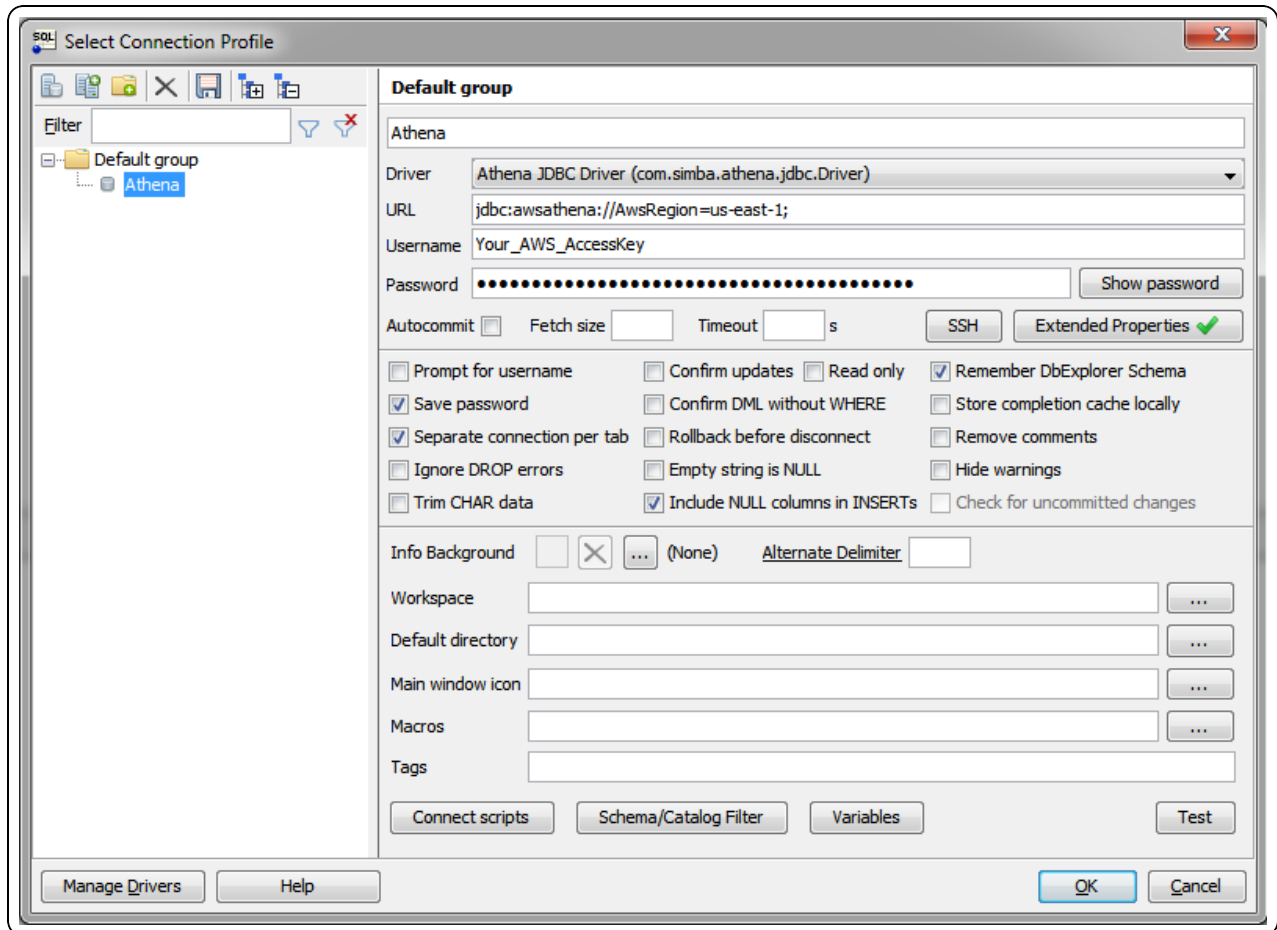
Field Name	Value
<b>Library</b>	The full path and name of the <code>AthenaJDBC [APIVersion].jar</code> file, where <code>[APIVersion]</code> is the JDBC version number that the driver supports.  For example, <b>AthenaJDBC42.jar</b> for the driver that supports JDBC 4.2.
<b>Classname</b>	<b>com.simba.athena.jdbc.Driver</b>
<b>Sample URL</b>	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, <b>jdbc:awsathena://AwsRegion=us-east-1;</b>



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
<b>URL</b>	<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, <b><code>jdbc:awsathena://AwsRegion=us-east-1;</code></b></p> <p>By default, this field is automatically populated with the Sample URL value that you specified for the selected driver.</p>
<b>Username</b>	The access key provided by your AWS account.
<b>Password</b>	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 27
- [Examples: Using a Credentials Provider](#) on page 27
- [Example: Executing a SELECT Query](#) on page 30
- [Example: Running a CREATE Statement](#) on page 30
- [Example: Listing Tables](#) on page 31

### 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("User", "AWSAccessKey");
info.put("Password", "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 28
- [Example: PropertiesFileCredentialsProvider](#) on page 28
- [Example: InstanceProfileCredentialsProvider](#) on page 28
- [Example: CustomSessionCredentialsProvider](#) on page 29

For more information about configuring the driver to authenticate your connection using a credentials provider, see [Using the AWSCredentialsProvider Interface](#) on page 33.

### 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.PropertiesFileCredentialsPr
ovider");
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.InstanceProfileCredentialsP
rovider");
```

### Example: CustomSessionCredentialsProvider

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

The credentials object can use either `BasicSessionCredentials` or `AWSCredentials`.

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 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;
    // AWSCredentials can also be used instead of
BasicSessionCredentials
    //Set aws_credentials_provider_class =
"com.amazonaws.custom.athena.jdbc.CustomIAMRoleAssumptionSAM
LCredentialsProvider"
    // set AwsCredentialsProviderArguments =
"<accessID>,<secretKey>,<sessionToken>"

    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(){
        //Use this method if refresh token
    }
}
```

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 33
- [Using the AWSCredentialsProvider Interface](#) on page 33
- [Using the Active Directory Federation Services \(AD FS\) Credentials Provider](#) on page 38
- [Using the Okta Credentials Provider](#) on page 44
- [Using a Custom Credentials Provider for an Identity Provider](#) on page 49

### 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 `User` property to the access key provided by your AWS account.
2. Set the `Password` 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.

 **Note:**

The AWS SDK is shaded and packaged in the driver JAR file. However, if your project has its own AWS SDK dependency, it is recommended that you use that class implementation instead of the one that is shaded in the driver. The shaded AWS SDK is intended to be used internally by the driver, and might not be optimal for other use cases.

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 34
- [Using PropertiesFileCredentialsProvider](#) on page 35
- [Using InstanceProfileCredentialsProvider](#) on page 35
- [Using a Custom Credentials Provider](#) on page 37

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 27.

## 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 28.

## 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 name 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 28.

## 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 examples of a few different ways that profiles can be used, see the following:

- [Example: Using a Profile to Provide a Session Token](#) on page 36
- [Example: Using a Profile to Switch Roles](#) on page 36

For a code example that demonstrates how to use the `InstanceProfileCredentialsProvider` in a Java application, see [Example: InstanceProfileCredentialsProvider](#) on page 28.

### Example: Using a Profile to Provide a Session Token

This example demonstrates how to authenticate a connection to Athena using a profile that provides a session token. A session token grants temporary access to the Athena service. An access key and secret key must be provided along with the token.

First, define a profile that specifies the access key, secret key, and session token. For example:

```
[simba_session]
aws_access_key_id=[YourAccessKey]
aws_secret_access_key=[YourSecretKey]
aws_session_token=[YourSessionToken]
```

Then, to connect to Athena using that profile, set the following properties in your connection URL:

- Set the `AwsCredentialsProviderClass` property to `com.simba.athena.amazonaws.auth.profile.ProfileCredentialsProvider`.
- Set the `AwsCredentialsProviderArguments` property to the name of the profile that you want to use. In this case, `simba_session`.

For example:

```
jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=s3://my-athena-resultbucket/test/;AwsCredentialsProviderClass=com.simba.athena.amazonaws.auth.profile.ProfileCredentialsProvider;AwsCredentialsProviderArguments=simba_session;
```

### Example: Using a Profile to Switch Roles

You can define a profile that refers to another existing profile, and then include an additional role setting. When you use this profile to authenticate your connection, the driver uses all the settings defined in the referenced profile as well as the role that has been additionally specified. By switching between these profiles, you can alternate between your roles without having to update the profile definition each time.

For an example that demonstrates how to configure profiles this way, see "Example Scenario: Switch to a Production Role" in the AWS Identity and Access Management documentation: [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_use\\_switch-role-cli.html#switch-role-cli-scenario-prod-env](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-cli.html#switch-role-cli-scenario-prod-env).

Based on this AWS example, which involves defining a main profile named `default` and a role switch profile named `prodaccess`, in order to switch roles when using the Simba Athena JDBC Driver, all you would need to do is update the `AwsCredentialsProviderArguments` setting in your connection URL. For example:

- To use the `ProductionAccessRole` role:

```
jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=s3://my-athena-resultbucket/test/;AwsCredentialsProviderClass=com.simba.athena.amazonaws.auth.profile.ProfileCredentialsProvider;AwsCredentialsProviderArguments=prodaccess;
```

- Or, to access Athena without using the `ProductionAccessRole` role:

```
jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=s3://my-athena-resultbucket/test/;AwsCredentialsProviderClass=com.simba.athena.amazonaws.auth.profile.ProfileCredentialsProvider;AwsCredentialsProviderArguments=default;
```

## 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.

For an example of using a custom credentials provider to obtain credentials from an identity provider, see [Using a Custom Credentials Provider for an Identity Provider](#) on page 49.

### 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](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 29.

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.

## Using the Active Directory Federation Services (AD FS) Credentials Provider

You can configure the driver to authenticate the connection using the AD FS credentials provider, which obtains credentials from the AD FS identity provider. To do this, connect to Athena using a connection URL that does either of the following:

- Includes property settings that specify information about the AD FS service. For more information, see [Specifying AD FS Information in the Connection URL](#) on page 39.
- Refers to an AWS profile that specifies information about the AD FS service. For more information, see [Specifying AD FS Information in an AWS Profile](#) on page 41.

### ! Important:

- If any information is included in both places, the information specified directly in the connection URL takes precedence over the information in the profile.
- If the connection URL refers to an AWS profile, then the `AWSCredentialsProviderClass` property must be specified in the profile instead of the connection URL.

When the driver connects to Athena, it retrieves temporary credentials from the AD FS identity provider. If these credentials are associated with an IAM role that has permission to access Athena, the driver immediately uses these credentials to authenticate the connection to Athena. Otherwise, you must exchange the temporary credentials for more specialized AWS credentials, which can then be used to authenticate the connection. For post-SAML workflows such as exchanging temporary

credentials for specialized ones, the driver provides a post-SAML workflow hook. For more information, see [Using the Post-SAML Workflow Hook](#) on page 57.

## Specifying AD FS Information in the Connection URL

In your connection URL, set properties to specify information such as the host and port of the server where the AD FS service is hosted.

If your connection URL also specifies an AWS profile that contains some AD FS information, then the settings specified directly in the URL take precedence over the AD FS information in the profile, and the `AWSCredentialsProviderClass` property must be specified in the profile instead of the connection URL.



### Note:

Some properties can be set through aliases, as described below. If you specify both a property name and its alias, the setting associated with the property name takes precedence.

### To specify AD FS information in the connection URL:

- In your connection URL, set the following properties:

Property	Value
<code>AWSCredentialsProviderClass</code> As alternatives, you can configure this property using the aliases <code>aws_credentials_provider_class</code> or <code>plugin_name</code> . If you specify both aliases, the setting associated with <code>aws_credentials_provider_class</code> takes precedence.	The FQCN that implements the AD FS credentials provider.
<code>IdP_Host</code>	The host name of the AD FS service that you are using to authenticate the connection.  The host name cannot include any slashes (/).

Property	Value
<p>IdP_Port</p>	<p>The number of the port that the AD FS service host uses to listen for requests.</p> <p>The exact port number that you need to specify may differ depending on the AD FS server configuration. If you are not sure which port to specify, contact your system administrator.</p>
<p>User</p> <p>As an alternative, you can configure this property using the alias <code>UID</code>.</p>	<p>The user name that you use to access the AD FS server. You can include the domain name using the format <code>[DomainName]\[UserName]</code>.</p> <p>On Windows machines, if you do not provide a user name, the driver attempts to authenticate to the AD FS server using your Windows user name over the NTLM protocol.</p>
<p>Password</p> <p>As an alternative, you can configure this property using the alias <code>PWD</code>.</p>	<p>The password corresponding to your user name that you specified in the <code>User</code> or <code>UID</code> property.</p> <p>On Windows machines, if you do not provide a password, the driver attempts to authenticate to the AD FS server using your Windows password over the NTLM protocol.</p>
<p>preferred_role</p>	<p>The Amazon Resource Name (ARN) of the role that you want to assume when authenticated through AD FS.</p>
<p>SSL_Insecure</p>	<p>One of the following:</p> <ul style="list-style-type: none"> <li>• <code>false</code> if you want the driver to verify the server certificate.</li> <li>• <code>true</code> if you do not want the driver to verify the server certificate.</li> </ul>

For example:

```
jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=
s3://test;AwsCredentialsProviderClass=com.simba.athena.
iamsupport.plugin.AdfsCredentialsProvider;idp_host=
example.adfs.server;idp_port=443;User=HOME\jsmith;
Password=simba12345;preferred_role=
arn:aws:iam::123456789123:role/JSMITH;ssl_insecure=true;
```

When you connect to Athena, the driver retrieves temporary credentials from AD FS. If these credentials are not associated with an IAM role that has permission to access Athena, then you must exchange them for more specialized AWS credentials before the driver can authenticate the connection. For information about how to complete this process, see [Using the Post-SAML Workflow Hook](#) on page 57.

## Specifying AD FS Information in an AWS Profile

In your AWS credentials file, define a profile that specifies information such as the host and port of the server where the AD FS service is hosted, and your credentials for accessing the AD FS service. Then, in your connection URL, set the `profile` property to the name of that profile.

By default, the AWS credentials file is located in `~/.aws/credentials`. You can change this default behavior by setting the `AWS_CREDENTIAL_PROFILES_FILE` environment variable to the full path and name of a different credentials file. For more information about profiles, see "Working with AWS Credentials" in the *AWS SDK for Java Developer Guide*: <https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html>.

If any AD FS information is also specified directly in your connection URL, those settings take precedence over the AD FS information in the profile.



### Note:

Some properties can be set through aliases, as described below. If you specify both a property name and its alias, the setting associated with the property name takes precedence.

### To specify AD FS information in an AWS profile:

1. In your AWS credentials file, define a profile that specifies the following property settings. Start by providing the name of the profile in brackets (`[ ]`), and then specify each property on separate lines.

Property	Value
<p><code>AWSCredentialsProviderClass</code></p> <p>As alternatives, you can configure this property using the aliases <code>aws_credentials_provider_class</code> or <code>plugin_name</code>. If you specify both aliases, the setting associated with <code>aws_credentials_provider_class</code> takes precedence.</p>	<p>The FQCN that implements the AD FS credentials provider.</p>
<p><code>IdP_Host</code></p>	<p>The host name of the AD FS service that you are using to authenticate the connection.</p> <p>The host name cannot include any slashes (/).</p>
<p><code>IdP_Port</code></p>	<p>The number of the port that the AD FS service host uses to listen for requests.</p> <p>You can specify any port that is available on your AD FS server. If you are not sure which port to specify, contact your system administrator.</p>
<p><code>User</code></p> <p>As an alternative, you can configure this property using the alias <code>UID</code>.</p>	<p>The user name that you use to access the AD FS server. You can include the domain name using the format <code>[DomainName]\[UserName]</code>.</p> <p>On Windows machines, if you do not provide a user name, the driver attempts to authenticate to the AD FS server using your Windows user name over the NTLM protocol.</p>

Property	Value
Password As an alternative, you can configure this property using the alias <code>PWD</code> .	The password corresponding to your user name that you specified in the <code>User</code> or <code>UID</code> property.  On Windows machines, if you do not provide a password, the driver attempts to authenticate to the AD FS server using your Windows password over the NTLM protocol.
<code>preferred_role</code>	The Amazon Resource Name (ARN) of the role that you want to assume when authenticated through AD FS.
<code>SSL_Insecure</code>	One of the following: <ul style="list-style-type: none"> <li>• <code>false</code> if you want the driver to verify the server certificate.</li> <li>• <code>true</code> if you do not want the driver to verify the server certificate.</li> </ul>

For example, the following is an AWS profile named `plug-in-creds-lambda` that specifies all the required AD FS service information:

```
[plug-in-creds-lambda]
plugin_name=com.providers.AdfsLambdaCredentialProvider
idp_host=example.adfs.server
idp_port=443
preferred_role=arn:aws:iam::123456789123:role/JSMITH
user=HOME\jsmith
password=simba12345
```

2. In your connection URL, set the `profile` property to the name of the profile.

For example:

```
jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=s3://test;profile=plug-in-creds-lambda;
```

When you connect to Athena, the driver checks the AWS credentials file for the specified profile, and then uses the AD FS information given in the profile to retrieve

temporary credentials from AD FS. If these credentials are not associated with an IAM role that has permission to access Athena, then you must exchange them for more specialized AWS credentials before the driver can authenticate the connection. For information about how to complete this process, see below.

## Using the Okta Credentials Provider

You can configure the driver to authenticate the connection using the Okta credentials provider, which obtains credentials from the Okta identity provider. To do this, connect to Athena using a connection URL that does either of the following:

- Includes property settings that specify information about the Okta service. For more information, see [Specifying Okta Information in the Connection URL](#) on page 44.
- Refers to an AWS profile that specifies information about the Okta service. For more information, see [Specifying Okta Information in an AWS Profile](#) on page 46.

### ! Important:

- If any information is included in both places, the information specified directly in the connection URL takes precedence over the information in the profile.
- If the connection URL refers to an AWS profile, then the `AWSCredentialsProviderClass` property must be specified in the profile instead of the connection URL.

When the driver connects to Athena, it retrieves temporary credentials from the Okta identity provider. If these credentials are associated with an IAM role that has permission to access Athena, the driver immediately uses these credentials to authenticate the connection to Athena. Otherwise, you must exchange the temporary credentials for more specialized AWS credentials, which can then be used to authenticate the connection. For post-SAML workflows such as exchanging temporary credentials for specialized ones, the driver provides a post-SAML workflow hook. For more information, see [Using the Post-SAML Workflow Hook](#) on page 57.

## Specifying Okta Information in the Connection URL

In your connection URL, set properties to specify information such as the host of the server where the Okta service is hosted.

If your connection URL also specifies an AWS profile that contains some Okta information, then the settings specified directly in the URL take precedence over the Okta information in the profile, and the `AWSCredentialsProviderClass` property must be specified in the profile instead of the connection URL.

 **Note:**

Some properties can be set through aliases, as described below. If you specify both a property name and its alias, the setting associated with the property name takes precedence.

**To specify Okta information in the connection URL:**

- In your connection URL, set the following properties:

Property	Value
IdP_Host	The host name of the Okta service that you are using to authenticate the connection.  The host name cannot include any slashes (/).
AWSCredentialsProviderClass  As alternatives, you can configure this property using the aliases <code>aws_credentials_provider_class</code> or <code>plugin_name</code> . If you specify both aliases, the setting associated with <code>aws_credentials_provider_class</code> takes precedence.	The FQCN that implements the Okta credentials provider.
App_ID	The Okta-provided unique ID associated with your Athena application.
App_Name	(Optional) The name of the Okta application that you use to authenticate the connection to Athena.
User  As an alternative, you can configure this property using the alias <code>UID</code> .	The email address that you use to access the Okta server.

Property	Value
Password As an alternative, you can configure this property using the alias <code>PWD</code> .	The password corresponding to your user name specified in the <code>User</code> or <code>UID</code> property.
<code>preferred_role</code>	The Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.
<code>SSL_Insecure</code>	One of the following: <ul style="list-style-type: none"> <li><code>false</code> if you want the driver to verify the server certificate.</li> <li><code>true</code> if you do not want the driver to verify the server certificate.</li> </ul>

For example:

```
jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=s3://test;AwsCredentialsProviderClass=com.simba.athena.iamsupport.plugin.OktaCredentialsProvider;UID=jsmith@acme.com;PWD=simba12345;idp_host=123456.okta.com;app_id=45L3MDWLKM4EDMWR34M;ssl_insecure=true;
```

When you connect to Athena, the driver retrieves temporary credentials from Okta. If these credentials are not associated with an IAM role that has permission to access Athena, then you must exchange them for more specialized AWS credentials before the driver can authenticate the connection. For information about how to complete this process, see [Using the Post-SAML Workflow Hook](#) on page 57.

## Specifying Okta Information in an AWS Profile

In your AWS credentials file, define a profile that specifies information such as the host of the server where the Okta service is hosted, and your credentials for accessing the Okta service. Then, in your connection URL, set the `profile` property to the name of that profile.

By default, the AWS credentials file is located in `~/.aws/credentials`. You can change this default behavior by setting the `AWS_CREDENTIAL_PROFILES_FILE` environment variable to the full path and name of a different credentials file. For more

information about profiles, see "Working with AWS Credentials" in the *AWS SDK for Java Developer Guide*: <https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html>.

If any Okta information is also specified directly in your connection URL, those settings take precedence over the Okta information in the profile.



**Note:**

Some properties can be set through aliases, as described below. If you specify both a property name and its alias, the setting associated with the property name takes precedence.

**To specify Okta information in an AWS profile:**

1. In your AWS credentials file, define a profile that specifies the following property settings. Start by providing the name of the profile in brackets ([ ]), and then specify each property on separate lines.

Property	Value
IdP_Host	The host name of the Okta service that you are using to authenticate the connection.  The host name cannot include any slashes (/).
AWSCredentialsProviderClass  As alternatives, you can configure this property using the aliases <code>aws_credentials_provider_class</code> or <code>plugin_name</code> . If you specify both aliases, the setting associated with <code>aws_credentials_provider_class</code> takes precedence.	The FQCN that implements the Okta credentials provider.
App_ID	The Okta-provided unique ID associated with your Athena application.

Property	Value
App_Name	(Optional) The name of the Okta application that you use to authenticate the connection to Athena .
User As an alternative, you can configure this property using the alias <code>UID</code> .	The email address that you use to access the Okta server.
Password As an alternative, you can configure this property using the alias <code>PWD</code> .	The password corresponding to your user name that you specified in the <code>User</code> or <code>UID</code> property.
preferred_role	The Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.
SSL_Insecure	One of the following: <ul style="list-style-type: none"> <li><code>false</code> if you want the driver to verify the server certificate.</li> <li><code>true</code> if you do not want the driver to verify the server certificate.</li> </ul>

For example, the following is an AWS profile named `plug-in-creds-okta` that specifies all the required Okta service information:

```
[plug-in-creds-okta]
plugin_
name=com.simba.athena.iamsupport.plugin.OktaCredentialsP
rovider
idp_host=123456.okta.com
app_id=45L3MDWLKM4EDMWR34M
uid=jsmith@acme.com
pwd=simba12345
```

2. In your connection URL, set the `profile` property to the name of the profile.

For example:

```
jdbc:awsathena://AwsRegion=us-east-1;S3OutputLocation=s3://test-athena-output-us-east-1/profile=plug-in-creds-okta;ssl_insecure=true;
```

When you connect to Athena, the driver checks the AWS credentials file for the specified profile, and then uses the Okta information given in the profile to retrieve temporary credentials from Okta. If these credentials are not associated with an IAM role that has permission to access Athena, then you must exchange them for more specialized AWS credentials before the driver can authenticate the connection. For information about how to complete this process, see below.

## Using a Custom Credentials Provider for an Identity Provider

You can configure the driver to use a custom credentials provider class that obtains credentials from an identity provider. To do this, set the `AwsCredentialsProviderClass` connection property to the FQCN of the credentials provider class, and set the `AwsCredentialsProviderArguments` connection property to a comma-separated list of any String arguments necessary for the constructor of that credentials provider class.

For example, if you are using a credentials provider class named `com.you.credential.provider.SampleSamlCredentialsProvider` and it requires String arguments named `KEY_SAML_ASSERTION` and `CUSTOM_PROVIDER_PERMISSIONS`, you would use the following connection URL to connect to Athena:

```
jdbc:awsathena://AwsRegion=us-east-1;AwsCredentialsProviderClass=com.you.credential.provider.SampleSamlCredentialsProvider;AwsCredentialsProviderArguments=KEY_SAML_ASSERTION,CUSTOM_PROVIDER_PERMISSIONS;
```

## Creating a Custom Credentials Provider Class for an Identity Provider

To create a custom credentials provider class for an identity provider, do one of the following:

- Extend the `SampleSamlCredentialsProvider` class provided by the driver to implement the procedures specific to your identity provider. Compared to the

other implementations in this list, this one is more closely integrated with the driver implementation, and includes support for the relevant Simba Athena JDBC Driver connection properties (such as `idp_host`, `idp_port`, `preferred_role`, and `ssl_insecure`). For more information, see [Extending the `SamlCredentialsProvider` Class for an Identity Provider](#) on page 50.

- Implement the `com.simba.iamsupport.IPlugin` interface to define the methods for working with your identity provider. This implementation requires you to define all the methods for interacting with the identity provider. Additionally, it provides access to the driver's connection properties through the `addParameter()` method. For more information, see [Implementing the `com.simba.iamsupport.IPlugin` Interface for an Identity Provider](#) on page 56.
- Define a class that extends the `com.amazonaws.auth.AWSCredentialsProvider` class to retrieve your AWS credentials from the provider. This implementation works similarly to the previous one in that it also requires you to define all the methods for interacting with the identity provider. However, this implementation does not support the `addParameter()` method. To pass in the parameters that the class requires, you must use the `AwsCredentialsProviderArguments` connection property instead. For more information, see [Extending the `com.amazonaws.auth.AWSCredentialsProvider` Class for an Identity Provider](#) on page 57.

## Extending the `SamlCredentialsProvider` Class for an Identity Provider

The Simba Athena JDBC Driver provides a class named `SamlCredentialsProvider`. You can extend this class to implement the procedures for an identity provider using the SAML standard, so that the driver can authenticate connections using the credentials from the provider.

The exact procedure for implementing these procedures depends on the requirements specific to your identity provider. The following is a high-level overview of what a typical implementation might involve:

1. Import the following libraries from the `com.simba.athena.iamsupport.plugin` package:
  - `java.io.IOException`
  - `com.simba.athena.iamsupport.plugin.SamlCredentialsProvider`

If your implementation requires the `performPostSAMLAction()` method, then also import the following libraries:

- `com.simba.athena.amazonaws.SdkClientException`
  - `com.simba.athena.iamsupport.model.CredentialsHolder`
2. Extend the `SamlCredentialsProvider` class, and include the following in your class definition:

- A definition for each custom connection property that the provider requires.
- A field for specifying the values for these custom connection properties.
- Optionally, a default constructor.
- An override for the `addParameter()` method, to specify if-conditions that check for custom connection properties and implements the necessary behavior. At minimum, this override should include the parent class that provides support for all the connection properties that the Simba Athena JDBC Driver supports by default.
- An override for the `getSamlAssertion()` method, to return the SAML assertion string from your provider to the driver. This enables the current class implementation to retrieve temporary credentials from the provider.
- A post-SAML workflow hook to exchange any temporary credentials for proper AWS credentials. For more information, see [Using the Post-SAML Workflow Hook](#) on page 57.

The following code sample shows how this might be accomplished:

```
package com.simba.athena.iamsupport.plugin;
import java.io.IOException;
import com.simba.athena.iamsupport.plugin.SamlCredentials
Provider;
/**
 * The following two imports are needed for the
 * performPostSAMLAction() method.
 * They are not needed if your implementation does not
 * require this method.
 */
import com.simba.athena.amazonaws.SdkClientException;
import com.simba.athena.iamsupport.model.CredentialsHolder;
/**
 * A basic SAML credentials provider class. This class can be
 * changed and implemented to work with any desired SAML
 * service provider.
 */
public class SampleSamlCredentialsProvider extends
SamlCredentialsProvider
{
    /**
     * Here we are defining a new connection property key
     * called "saml_assertion". This property is specific
     * to the SampleSamlCredentialsProvider, and is used
     * to provide some information through the connection
     * URL.
     */
}
```

```
* <p>
* For example, when using this credentials provider,
* you may include the following in the connection URL:
* <p>
* <code>
* jdbc:awsathena://AwsRegion=[Region];
* S3OutputLocation=[Output];saml_assertion=[Value]
* </code>
* <p>
* If your implementation requires user input through
* the connection URL, you can define the key name of
* the property using the following code:
* <p>
* <code>
* public static final String PROPERTY_NAME =
* "key_name";
* </code>
* <p>
* You can add as many new connection properties as
* needed.
* "key_name" is subject to the following restrictions:
* <p>
* - The name must be unique. It cannot match any
* existing connection property key name in the
* Simba Athena JDBC Driver. These key names are
* case-insensitive, so even if the case does not match
* what is found in the documentation, the key name is
* not allowed.
* <p>
* - The key name cannot contain any spaces.
* <p>
* - The key name can only contain letters of the
* alphabet (upper-case and lower-case) or
* underscores (_).
*
* /
public static final String KEY_SAML_ASSERTION = "saml_
assertion";
/**
 * This field stores the value given with the
 * associated connection property key.
```

```
    * <p>
    * If you add more connection property keys, then you
    * must define additional fields to hold those values.
    */
private String samlAssertion;
/**
 * Optional default constructor.
 */
public SampleSamlCredentialsProvider()
{
}
/**
 * This method is used to get the values associated
 * with different connection string properties.
 * <p>
 * We override it in this custom credentials provider
 * to add a check for any additional connection
 * properties that were added, which are not included
 * in the existing Athena JDBC driver. It allows us
 * to store these values using the appropriate fields
 * as mentioned above.
 * <p>
 * For any new connection property keys added to this
 * class, add an if-condition to check if the current
 * key matches the connection property key. If so,
 * store the value associated with the key in the
 * appropriate field.
 * <p>
 * If no new connection property keys are required, you
 * may leave the implementation blank and simply return
 * a call to the parent class implementation.
 * <p>
 * See the example below.
 *
 * @param key A string representing the connection
 * property key.
 * @param value The value associated with the
 * connection property key.
 */
@Override
public void addParameter(String key, String value)
```

```
{
    // The parent class takes care of setting up all the
    // connection properties that are mentioned in the
    // Simba Athena JDBC Driver documentation.
    super.addParameter(key, value);

    // Add if-condition checks for any connection
    // properties which are specific to your
    // implementation of this custom SAML credentials
    // provider.
    if (KEY_SAML_ASSERTION.equalsIgnoreCase(key))
    {
        samlAssertion = value;
    }
}
/**
 * This method needs to return the SAML assertion
 * string returned by the specific SAML provider
 * being used for this implementation. The exact
 * instructions for getting this string vary
 * depending on the specific SAML provider you
 * are using.
 * <p>
 * This is used by the SamlCredentialsProvider parent
 * class to get the temporary credentials.
 *
 * @return The SAML assertion string.
 * @throws IOException
 */
@Override
protected String getSamlAssertion() throws IOException
{
    /**
     * If you wish to make a connection property required,
     * you can check that the associated field has been
     * populated, and if not, throw an IOException.
     * if (StringUtil.isNullOrEmpty(samlAssertion))
     * {
     * throw new IOException("Missing required property: "
     + KEY_SAML_ASSERTION);
     * }
     */
}
```

```
    */
    return samlAssertion;
}

/**
 * This method is used to perform custom actions after
 * temporary credentials become available.
 * This method is only needed if your authentication
 * workflow requires an additional step to turn
 * temporary credentials into AWS credentials.
 *
 * The inputs are the temporary credentials returned by
 * the provider, and a SAML assertion that determines
 * how these credentials are exchanged for AWS
 * credentials.
 *
 * @param username The user name.
 * @param samlAssertion The SAML assertion.
 * @param credentials The temporary credentials from
 * the assumeRoleWithSAML request.
 *
 * @return The updated or new credentials from the
 * post-SAML authentication action.
 * If null is returned, this indicates that no
 * post-SAML authentication action was performed and
 * the
 * existing credentials should not be updated.
 *
 * @throws SdkClientException The exception that
 * occurred.
 */
protected CredentialsHolder performPostSAMLAction(
    String username,
    String samlAssertion,
    CredentialsHolder credentials) throws
    SdkClientException
{
    return null;
}
}
```

## Implementing the `com.simba.iamsupport.IPlugin` Interface for an Identity Provider

You can define a custom class that implements the `com.simba.iamsupport.IPlugin` interface to define the methods for working with an identity provider.

The exact procedure for doing this depends on the specific requirements for your identity provider. In general, you need to define all the methods for interacting with the identity provider. You also have the option of using the `addParameter()` method to define custom connection properties.

The following example shows how to use `addParameter()`:

```
/**
 * This method is used to get the values associated with
 * different connection string properties.
 * <p>
 * We override it in this custom credentials provider to
 * add a check for any additional connection properties
 * that were added, which are not included in the existing
 * Athena JDBC driver. It allows us to store these values
 * using the appropriate fields as mentioned above.
 * <p>
 * For any new connection property keys added to this
 * class, add an if-condition to check for the following:
 * if the current key matches the connection property key,
 * store the value associated with the key in the
 * appropriate field.
 * <p>
 * If no new connection property keys are required, you
 * may leave the implementation blank and simply return
 * a call to the parent class implementation.
 * <p>
 * See the example below.
 *
 * @param key A string representing the connection
 * property key.
 * @param value The value associated with the connection
 * property key.
 */
@Override
public void addParameter(String key, String value)
```

```
{
    // The parent class takes care of setting up all the
    // connection properties that are mentioned in the Athena
    // JDBC driver documentation.
    super.addParameter(key, value);

    // Add if-condition checks for any connection properties
    // which are specific to your implementation of this
    // custom SAML credentials provider.
    if (KEY_SAML_ASSERTION.equalsIgnoreCase(key))
    {
        samlAssertion = value;
    }
}
```

## Extending the `com.amazonaws.auth.AWSCredentialsProvider` Class for an Identity Provider

You can extend the `com.amazonaws.auth.AWSCredentialsProvider` class to retrieve AWS credentials from an identity provider.

The exact procedure for doing this depends on the specific requirements for your identity provider. In general, you need to define all the methods for interacting with the identity provider.

Be aware that you cannot use `addParameter()` to define custom connection properties. Instead, you must use the `AWSCredentialsProviderArguments` connection property to pass in all the parameters that this class requires. For more information about this property, see [AwsCredentialsProviderClass](#) on page 78.

For an example of a class that extends the `com.amazonaws.auth.AWSCredentialsProvider` class and uses the `AwsCredentialsProviderArguments` to pass in parameters, see [Example: CustomSessionCredentialsProvider](#) on page 29.

## Using the Post-SAML Workflow Hook

In some cases, after retrieving credentials through a SAML workflow, you may need to modify the credentials before they can actually be used to authenticate your connection to Athena. For example, if you retrieve temporary credentials from an AD FS provider, and these credentials are not associated with an IAM role that has permission to access Athena, you must exchange them for more specialized

AWS credentials before you can authenticate the connection. The driver provides a post-SAML workflow hook that enables you to go through such processes.

To exchange the temporary credentials from AD FS for more specialized AWS credentials, do the following:

1. Extend the default AD FS credentials provider class. The FQCN of this class is:

```
com.simba.athena.iamsupport.plugin.AdfsCredentialsProvider
```

2. Override the `performPostSAMLAction()` method with the post-SAML workflow hook. For details, see below.



**Note:**

For a complete implementation example, see [Example: Implementing a Custom Credentials Provider that uses the Post-SAML Workflow Hook](#) on page 60.

To override the `performPostSAMLAction()` method, use a post-SAML workflow hook that includes your AD FS user name, the temporary credentials returned by the AD FS service, and a SAML assertion that determines how these credentials are exchanged for AWS credentials.

The function signature of the hook is as follows:

```
/**
 * Perform custom actions after the temporary credentials
 * are available.
 *
 * @param username      Your AD FS user name.
 * @param samlAssertion The Base64-encoded SAML
 *                      assertion.
 * @param credentials   The temporary credentials from
 *                      the AssumeRoleWithSAML request.
 *
 * @return              The CredentialsHolder wrapper
 *                      object containing the AWS
 *                      credentials.
 */
@Override
protected CredentialsHolder performPostSAMLAction(
    String username,
    String samlAssertion,
    CredentialsHolder credentials) throws
    SdkClientException
```

**! Important:**

It is recommended that you specify `@Override`, as this enables the build to return an error if a class mismatch occurs.

This implementation causes the following to occur:

1. The AD FS credentials provider class obtains the following:
  - The SAML assertion from the AD FS provider.
  - The temporary credentials with the `AssumeRoleWithSAML` API from STS.
2. The specified AD FS user name, SAML assertion, and temporary credentials are passed into the `performPostSAMLAction()` method.
3. The method returns a `CredentialsHolder` wrapper object containing your AWS credentials for authenticating your connection to Athena.

**Note:**

If the method returns NULL, this indicates that your credentials were not exchanged. This can occur if you pass in credentials that do not need to be exchanged and can be used immediately for authentication. If NULL is being returned unexpectedly, make sure that the post-SAML workflow hook is implemented properly and that the correct parameter values are being passed in.

4. The driver authenticates the connection to Athena using the credentials returned by the implementation.

The FQCN of the `CredentialsHolder` is:

```
com.simba.athena.iamsupport.model.CredentialsHolder
```

The `CredentialsHolder` can optionally include an expiration date for the returned credentials. If the expiration date is not included, then the function signature is as follows:

```
/**
 * Creates a new instance of the CredentialsHolder.
 *
 * @param credentials    The AWS credentials.
 *
 * @return              The CredentialsHolder.
 */
public static CredentialsHolder newInstance(AWSCredentials
credentials)
```

If the expiration date is included, then the function signature is as follows:

```
/**
 * Creates a new instance of the CredentialsHolder.
 *
 * @param credentials      The AWS credentials.
 * @param expiration       The expiration date.
 *
 * @return                 The CredentialsHolder.
 */
public static CredentialsHolder newInstance(AWSCredentials
credentials, Date expiration)
```

## Example: Implementing a Custom Credentials Provider that uses the Post-SAML Workflow Hook

The following Java example implements the AD FS Lambda credentials provider with the post-SAML workflow hook. The AWS Lambda service exchanges the temporary credentials for AWS credentials, which are returned to the driver and are then used to access Athena.

```
import com.simba.athena.iamsupport.model.CredentialsHolder;
import com.simba.athena.iamsupport.plugin.AdfsCredentialsProvider;

/**
 * The ADFS Lambda Credential Provider.
 */
public class AdfsLambdaCredentialProvider extends AdfsCredentialsProvider
{
    /**
     * Perform custom actions after temporary credentials are available.
     *
     * @param username      Your AD FS user name.
     * @param samlAssertion The Base64-encoded SAML assertion.
     * @param credentials   The temporary credentials from the assumeRoleWithSAML
     *                      request.
     *
     * @return The CredentialsHolder wrapper object that holds the new credentials.
     *         Returning null if no post-SAML action is performed.
     */
    @Override
    protected CredentialsHolder performPostSAMLAction(
        String username,
        String samlAssertion,
        CredentialsHolder credentials) throws SdkClientException
    {
        // Perform post-SAML work flow here.
        //
        //
        // AWSLambda client;
        try
        {
            // === Example:===

```

```

// === Temporary Credentials from ADFS to BasicSessionCredentials: ===
//
AWSCredentials cred = new BasicSessionCredentials(
    credentials.getAccessKeyId(),
    credentials.getSecretAccessKey(),
    credentials.getSessionToken());

// === Example===
// === Example of a custom credentials provider communicating against a
// === Lambda server with the temporary credentials acquired from STS
// === through the AD FS workflow. Then, it calls AthenaIAMRoleTest, a
// === custom Lambda function implemented on the AWS Lambda service,
// === using the temporary credentials to acquire a new set of
// === credentials from the Lambda function. Finally, it returns the new
// === credentials wrapped with CredentialsHolder class back to the
// === AdfsCredentialsProvider to consume.
//
// // Building a lambda client.
// AWSCredentialsProvider basic = new AWSStaticCredentialsProvider(cred);
// AWSLambdaClientBuilder builder = AWSLambdaClientBuilder
//     .standard()
//     .withRegion(Regions.US_EAST_1)
//     .withCredentials(basic);

// client = builder.build();
//
// Result result;
//
// // Serialize a payload object and send it to the Lambda function
// // AthenaIAMRoleTest.
// Payload p = new Payload(username, samlAssertion, true);
// String payload = s_objectMapper.writeValueAsString(p);
// System.out.println("PAYLOAD -> " + payload);
// InvokeRequest invReq = new InvokeRequest()
//     .withFunctionName("AthenaIAMRoleTest")
//     .withPayload(payload)
//     .withRequestCredentialsProvider(basic);
//
// InvokeResult fnret = client.invoke(invReq);
// String err = fnret.getFunctionError();
// if (err != null)
// {
//     System.err.println("ERROR -> " + err);
//     // Then we should throw an exception.
// }
// else
// {
//     // Process new credentials/roles returned by the Lambda function.
//     ByteBuffer bb = fnret.getPayload();
//     String data = CHARSET.decode(bb).toString();
//     System.out.println("DATA -> " + data);
//     JsonNode actualObj = s_objectMapper.readTree(data);
//     result = new Result(actualObj);
//     return CredentialsHolder.newInstance(result.getCredentials());
// }

// === Return the new credentials back to the driver: ===
//

```

```
        // return CredentialsHolder.newInstance(  
        //     new BasicAWSCredentials(  
        //         newAccessKey,  
        //         newSecretKey));  
        //  
        //  
        // === Or, if the new credentials have expiration information ===  
        // === Creates a new instance of the CredentialsHolder. ===  
        // === @param credentials    The AWSCredentials. ===  
        // === @param expiration    The credential expiration date. ===  
        // === @return                The CredentialsHolder. ===  
        //  
        // return CredentialsHolder.newInstance(  
        //     <AWSCredentials> credentials,  
        //     <Date> expiration);  
    }  
    catch (SdkClientException e)  
    {  
        // === Exception handling. ===  
    }  
}
```

## 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.

### ! Important:

If you are connecting to Athena through a proxy server, make sure that the proxy server does not block port 444. The result set streaming API uses port 444 on the Athena server for outbound communications. For more information, see [UseResultSetStreaming](#) on page 98.

### 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.

The settings for logging apply to every connection that uses the Simba Athena JDBC Driver, so make sure to disable the feature after you are done using it.

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;User=ABCABCABC123ABCABC45;Password=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.

## Configuring Connection Timeouts and Retries

You can configure the driver to wait for connections to respond and when to reattempt establishing a connection, to help automate the connection process.

### To configure a connection with timeout and retries:

1. Set the `ConnectionTimeout` property to amount of time, in seconds, that the driver should wait when establishing a connection before timing out the connection.
2. Set the `MaxErrorRetry` property to 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.

### Example: Connection URL with basic proxy connection properties defined:

```
jdbc:awsathena://AwsRegion=NA;User=1jt9Pcyq8pr3lvu143pfl4r86;Password=1jt9Pcyq8pr3lvu143pfl4r86;S3OutputLocation=SSE_S3;ConnectionTimeout=90;MaxErrorRetry=10;
```

## Features

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

- [Catalog and Schema Support](#) on page 69
- [File Formats](#) on page 69
- [Fetch Size](#) on page 69
- [Query Execution Polling](#) on page 70
- [Data Types](#) on page 71
- [Query ID Retrieval](#) on page 73
- [Integration with AWS Glue](#) on page 74
- [Security and Authentication](#) on page 74

### 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 92.

 **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.

## Query Execution Polling

When a query is run, the driver polls the Athena server for the query results until they are returned. The driver starts by polling the server frequently, and then increasing the interval of time between polls (decreasing the polling rate) as the query continues to run. You can configure the polling rate of the driver by setting the following properties:

- [MinQueryExecutionPollingInterval](#) on page 87
- [MaxQueryExecutionPollingInterval](#) on page 85
- [QueryExecutionPollingIntervalMultiplier](#) on page 92

The driver starts by polling the server every minimum interval, which is the number of milliseconds specified in `MinQueryExecutionPollingInterval`. The driver then increases the polling interval using the multiplier specified in `QueryExecutionPollingIntervalMultiplier` until the interval specified in `MaxQueryExecutionPollingInterval` is reached. The driver then continues to poll the server using this maximum interval until the query results are returned.

For example, the driver uses the following default settings for query execution polling:

- `MinQueryExecutionPollingInterval=5`
- `MaxQueryExecutionPollingInterval=1800000`
- `QueryExecutionPollingIntervalMultiplier=2`

Using these settings, the driver polls the server 5ms after the query begins to run, and then doubles the interval between polls after each subsequent poll. In other words, the driver polls the server at these intervals: 5ms after the query starts to run, 10ms after


the first poll, 20ms after the second poll, and so on until the 1800000ms polling interval is reached. The driver then continues to poll the server every 1800000ms until the query results are returned.

## 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 <a href="#">UseArraySupport</a> on page 97)	java.sql.Array of strings or string
BIGINT	BIGINT	long
BINARY	VARBINARY	byte[]
BOOLEAN	BOOLEAN	boolean
CHAR	CHAR	string
DATE	DATE	java.sql.Date
DECIMAL (p, s)	DECIMAL	java.math.BigDecimal
DOUBLE	DOUBLE	double
FLOAT	REAL	float

Athena Type	JDBC Type	Java Type
INTEGER <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <b>Note:</b>              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 <a href="#">Integer Support</a> on page 72.           </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>.

## Query ID Retrieval

To support the retrieval of query IDs from the Athenaserver, the driver provides an interface called `com.interfaces.core.IStatementQueryInfoProvider`. This interface provides the following functions:

- `public String getPreparedQueryId();`  
This function gets the ID of the query used when the statement was prepared. It returns the prepared query ID if available, or a null value if the ID is unavailable.
- `public String getQueryId();`  
This function gets the ID of the executed query. It returns the query ID if available, or a null value if the ID is unavailable.

The statement class used by the driver implements this interface, so these functions can be accessed if you unwrap the Java Statement object to this interface.

The following example shows how a `Statement` object might be used:

```
// Import the IStatementQueryInfoProvider interface
// Create the connection as normal
Statement stmt = conn.createStatement();
ResultSet result = stmt.executeQuery("SELECT * FROM
sen.integer_table");
String queryId = stmt.unwrap
(IStatementQueryInfoProvider.class).getQueryId(); // Get
query id after a statement has been executed.
```

The following example shows how a `PreparedStatement` object might be used:

```
// Import the IStatementQueryInfoProvider interface
// Create the connection as normal
PreparedStatement prepStmt = conn.prepareStatement("SELECT *
FROM sen.integer_table");
```

```
String prepQueryId = prepStmt.unwrap  
(IStatementQueryInfoProvider.class).getPreparedQueryId();  
prepStmt.execute();  
String queryId = prepStmt.unwrap  
(IStatementQueryInfoProvider.class).getQueryId();
```

## 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.

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](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 33.

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](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 63.

## 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 17.

 **Note:**

Property values are case-sensitive.

### App\_ID

Default Value	Data Type	Required
None	String	Yes, if authenticating through the Okta service.

#### Description

The Okta-provided unique ID associated with your Athena application.

### App\_Name

Default Value	Data Type	Required
None	String	No

#### Description

The name of the Okta application that you use to authenticate the connection to Athena.

## AwsCredentialsProviderArguments

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

### Description

#### Note:

As an alternative, you can configure this property using the alias `aws_credentials_provider_arguments`. If you specify both the property name and the alias for the same connection, the setting associated with the property name (`AwsCredentialsProviderArguments`) takes precedence.

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 33.

## AwsCredentialsProviderClass

Default Value	Data Type	Required
None	String	Yes, if <code>User</code> and <code>Password</code> are not provided, or if you are authenticating through AD FS or Okta.

### Description

 **Note:**

- As alternatives, you can configure this property using the aliases `aws_credentials_provider_class` or `plugin_name`. If you specify the property name and an alias for the same connection, the setting associated with the property name (`AwsCredentialsProviderArguments`) takes precedence. If you specify both aliases, the setting associated with `aws_credentials_provider_class` takes precedence.
- If the `Profile` property is set, then you must specify `AwsCredentialsProviderArguments` (or its aliases) in the profile instead of the connection URL.

This property is used differently, depending on your authentication configuration:

- If you are authenticating through AD FS or Okta, then set this property to the FQCN of the appropriate credentials provider. You can set this property in the connection URL or in an AWS profile.
- If you are authenticating through a class that implements the `AWSCredentialsProvider` interface, then set this property to the FQCN of the `AWSCredentialsProvider` interface.

 **Note:**

The AWS SDK is shaded and packaged in the driver JAR file. However, if your project has its own AWS SDK dependency, it is recommended that you use that class implementation instead of the one that is shaded in the driver. The shaded AWS SDK is intended to be used internally by the driver, and might not be optimal for other use cases.

## AwsRegion

Default Value	Data Type	Required
None	String	Yes, if an endpoint is not provided in the connection URL.

### 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.

For example, for the following endpoint URLs:

- `athena.[Region].amazonaws.com:443`
- `[vpce-specific-url].athena.[Region].vpce.amazonaws.com:443`

The driver parses the region from the endpoint and uses it to connect to Athena and AWS Glue services. Do not use the `AwsRegion` property if the endpoint is present in the connection string, because the `AwsRegion` value takes precedence over the region parsed from the endpoint.

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 verifies the connection by sending a “SELECT 1” query when 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 does not verify the connection. The connection string may contain unverified configuration values, such as incorrect authentication information, which will not be discovered at connection time. 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

### Note:

As an alternative, you can configure this property using the alias `connection_timeout`. If this alias is used, the amount of time is measured in milliseconds.

If you specify both the property name and the alias for the same connection, the setting associated with the property name (`ConnectTimeout`) takes precedence.

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.

## EndpointOverride

Default Value	Data Type	Required
None	String	No

## Description

The endpoint for the Athena instance that you want the driver to connect to. If you specify `AwsRegion` and connect to the default Athena instance, then you do not need to specify the endpoint.

The endpoint can also be specified directly in the connection URL. If endpoints are specified in both the connection URL and this property, the endpoint specified in this property takes precedence.

If you specify an override endpoint that does not follow the default endpoint format or the VPC endpoint format, the driver cannot parse it for the region. In this case corresponding region must be defined in the `AwsRegion` property.

For more information about endpoint overrides, see [Building the Connection URL](#) on page 17.

## IdP\_Host

Default Value	Data Type	Required
None	String	Yes, if authenticating through AD FS or Okta.

### Description

The host name of the AD FS or Okta service that you use to authenticate the connection. The host name cannot include any slashes (/).

You can set this property in the connection URL or in an AWS profile.

## IdP\_Port

Default Value	Data Type	Required
443	String	No

### Description

The number of the port that the AD FS service host uses to listen for requests.

The port number to specify may differ depending on the AD FS server configuration. If you are not sure which port to specify, contact your system administrator.

You can set this property in the connection URL or in an AWS profile.

## 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.

The settings for logging apply to every connection that uses the Simba Athena JDBC Driver, so make sure to disable the feature after you are done using it.

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 [UseAwsLogger](#) on page 97.

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.

 **Note:**

To make sure that the connection URL is compatible with all JDBC applications, it is recommended that you escape the backslashes (\) in your file path by typing another backslash.

## 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

 **Note:**

As an alternative, you can configure this property using the alias `max_error_retries`. If you specify both the property name and the alias for the same connection, the setting associated with the property name (`MaxErrorRetry`) takes precedence.

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.

This value must be a positive integer.

## MaxQueryExecutionPollingInterval

Default Value	Data Type	Required
1800000	Integer	No

### Description

The maximum amount of time, in milliseconds, that the driver waits between attempts when polling the Athena server for query results. You cannot specify an interval that is less than the interval specified in `MinQueryExecutionPollingInterval`.

For detailed information about the query execution polling feature, see [Query Execution Polling](#) on page 70.

## 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.

## MaxStreamErrorRetry

Default Value	Data Type	Required
5	Integer	No

### Description

The maximum number of attempts the driver makes to fetch a chunk from the streaming API.

This value must be a positive integer.

## 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.

## MinQueryExecutionPollingInterval

Default Value	Data Type	Required
5	Integer	No

### Description

The minimum amount of time, in milliseconds, that the driver waits between attempts when polling the Athena server for query results. You must specify a value that is greater than 0 but less than the value specified in

`MaxQueryExecutionPollingInterval`.

For detailed information about the query execution polling feature, see [Query Execution Polling](#) on page 70.

## 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.*
```

## Password

Default Value	Data Type	Required
None	String	Yes, if using any of the following for authentication: <ul style="list-style-type: none"> <li>• IAM credentials</li> <li>• AD FS</li> <li>• Okta</li> </ul>

## Description

### Note:

As an alternative, you can configure this property using the alias `PWD`. If you specify both the property name and the alias for the same connection, the setting associated with the property name (`Password`) takes precedence.

If you are using IAM credentials for authentication, then set this property to the secret key provided by your AWS account.

If you are authenticating through the AD FS credentials provider, then set this property to the password that you use to access the AD FS server. You can set this property in the connection URL or in an AWS profile. On Windows machines, if you do not provide a password, the driver attempts to authenticate to the AD FS server using your Windows password over the NTLM protocol.

If you are authenticating through the Okta credentials provider, then set this property to the password that you use to access the Okta server.

## 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.

## preferred\_role

Default Value	Data Type	Required
None. However, by default, the driver assumes the first role from the list returned in the SAML response from the identity provider.	String	No

### Description

The Amazon Resource Name (ARN) of the role that you want to assume when authenticated through AD FS or Okta.

You can set this property in the connection URL or in an AWS profile.

### Profile

Default Value	Data Type	Required
None	String	No

### Description

The name of the AWS profile to use, containing any additional connection properties not specified in the connection URL. For example, when configuring the driver to authenticate through AD FS, you can use this property to specify a profile that contains the required AD FS service information.

The driver checks the AWS credentials file for the specified profile. The default location for this file is `~/.aws/credentials`. You can change this default behavior by setting the `AWS_CREDENTIAL_PROFILES_FILE` environment variable to the full path and name of a different credentials file.

For more information about profiles, see "Working with AWS Credentials" in the *AWS SDK for Java Developer Guide*: <https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html>.

## 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.

## QueryExecutionPollingIntervalMultiplier

Default Value	Data Type	Required
2	Integer	No

### Description

The multiplier by which the driver increases the amount of time between polls, when polling the Athena server for query results. You cannot specify a value that is less than 2.

For detailed information about the query execution polling feature, see [Query Execution Polling](#) on page 70.

## 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 98 for details.

## S3OutputEncKMSKey

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

### Description

 **Note:**

As an alternative, you can configure this property using the alias `query_results_aws_kms_key`. If you specify both the property name and the alias for the same connection, the setting associated with the property name (`S3OutputEncKMSKey`) takes precedence.

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>.

## S3OutputEncOption

Default Value	Data Type	Required
None	String	No

### Description

 **Note:**

As an alternative, you can configure this property using the alias `query_results_encryption_option`. If you specify both the property name and the alias for the same connection, the setting associated with the property name (`S3OutputEncOption`) takes precedence.

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>.

## S3OutputLocation

Default Value	Data Type	Required
None	String	Yes, if the Workgroup property specifies a workgroup that is not configured with an output location.

## Description

**Note:**

As an alternative, you can configure this property using the alias `s3_staging_dir`. If you specify both the property name and the alias for the same connection, the setting associated with the property name (`S3OutputLocation`) takes precedence.

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`.

## 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

 **Note:**

As an alternative, you can configure this property using the alias `socket_timeout`. If this alias is used, the amount of time is measured in milliseconds.

If you specify both the property name and the alias for the same connection, the setting associated with the property name (`SocketTimeout`) takes precedence.

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.

## SSL\_Insecure

Default Value	Data Type	Required
false	String	No

### Description

This property indicates whether the server certificate of the AD FS host or the Okta service should be verified.

- `true`: The driver does not check the authenticity of the server certificate.
- `false`: The driver checks the authenticity of the server certificate.

You can set this property in the connection URL or in an AWS profile.

## StreamingEndpointOverride

Default Value	Data Type	Required
None	String	No

### Description

The endpoint for the Athena streaming service corresponding to the specified Athena instance. If this property is not set and `UseResultsetStreaming` is set to 1, the driver tries to connect to port 444 of the Athena instance endpoint specified in the connection string.

## 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.

## UseAwsLogger

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 66.

- 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.

## User

Default Value	Data Type	Required
None	String	Yes, if using any of the following for authentication: <ul style="list-style-type: none"> <li>• IAM credentials</li> <li>• AD FS</li> <li>• Okta</li> </ul>

## Description

### Note:

As an alternative, you can configure this property using the alias `UID`. If you specify both the property name and the alias for the same connection, the setting associated with the property name (`User`) takes precedence.

If you are using IAM credentials for authentication, then set this property to the access key provided by your AWS account.

If you are authenticating through the AD FS credentials provider, then set this property to the user name that you use to access the AD FS server. You can include the domain name using the format `[DomainName]\[UserName]`. You can set this property in the connection URL or in an AWS profile. On Windows machines, if you do not provide a user name, the driver attempts to authenticate to the AD FS server using your Windows user name over the NTLM protocol.

If you are authenticating through the Okta credentials provider, then set this property to the email address that you use to access the Okta server.

## 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.

### ! Important:

If you are connecting to Athena through a proxy server, make sure that the proxy server does not block port 444. The result set streaming API uses port 444 on the Athena server for outbound communications.

## Workgroup

Default Value	Data Type	Required
Primary	String	No

## Description

The workgroup you want your application to use when connecting.

## 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 Or 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	<b>1.x</b>	<b>2.x</b>
		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	<b>Metadata</b>	<b>Value</b>	<b>Metadata</b>	<b>Value</b>
	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	<b>Metadata</b>	<b>Value</b>	<b>Metadata</b>	<b>Value</b>
	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	<b>Metadata</b>	<b>Value</b>	<b>Metadata</b>	<b>Value</b>
	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	<b>Metadata</b>	<b>Value</b>	<b>Metadata</b>	<b>Value</b>
	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	<b>Metadata</b>	<b>Value</b>	<b>Metadata</b>	<b>Value</b>
	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	<b>Metadata</b>	<b>Value</b>	<b>Metadata</b>	<b>Value</b>
	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	<b>Metadata</b>	<b>Value</b>	<b>Metadata</b>
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 do the following:

- 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>.
- If you are connecting to Athena through a proxy server, make sure that the proxy server does not block port 444. The result set streaming API uses port 444 on the Athena server for outbound communications.

## 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.