

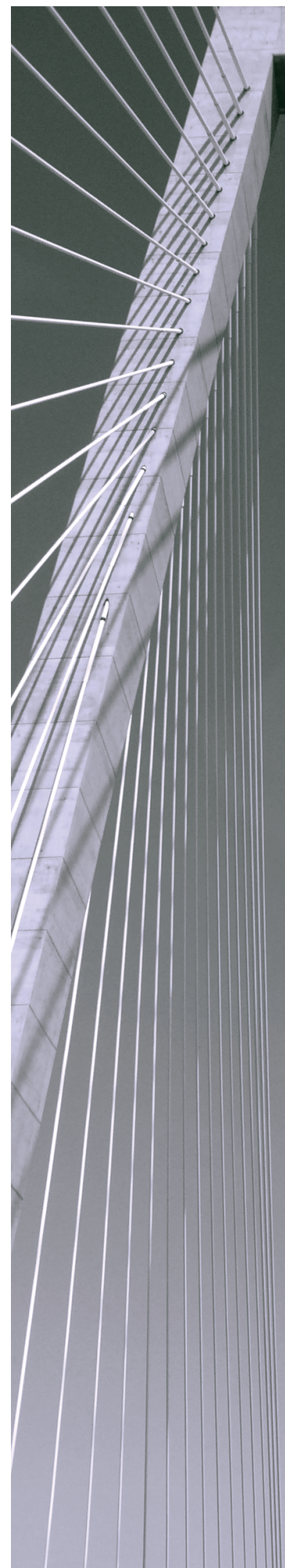


# Amazon JDBC Driver for Apache Hive

## Installation and Configuration Guide

Amazon Web Services, Inc.

September 15, 2015



**Copyright © 2015 Amazon Web Services, Inc. All Rights Reserved.**

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

Parts of this Program and Documentation include proprietary software and content that is copyrighted and licensed by Simba Technologies Incorporated. This proprietary software and content may include one or more feature, functionality or methodology within the ODBC, JDBC, ADO.NET, OLE DB, ODBO, XMLA, SQL and/or MDX component(s).

For information about Simba's products and services, visit: [www.simba.com](http://www.simba.com).

**Trademarks**

Simba, the Simba logo, SimbaEngine, SimbaEngine C/S, SimbaExpress and SimbaLib are registered trademarks of Simba Technologies Inc. All other trademarks and/or servicemarks are the property of their respective owners.

**Contact Us**

For support, check the EMR Forum at <https://forums.aws.amazon.com/forum.jspa?forumID=52> or open a support case using the AWS Support Center at <https://aws.amazon.com/support>

**Apache Hive**

Copyright 2008-2011 The Apache Software Foundation.

**Apache Thrift**

Copyright 2006-2010 The Apache Software Foundation.

**Apache ZooKeeper**

Copyright 2009-2012 The Apache Software Foundation.

**Apache log4j**

Copyright 1999-2014 The Apache Software Foundation.

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

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

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

**Simple Logging Façade for Java (SLF4J)**

Copyright (c) 2004-2013

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## About This Guide

### Purpose

The *Amazon JDBC Driver for Apache Hive Installation and Configuration Guide* explains how to install and configure the Amazon JDBC Driver for Apache Hive 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 Amazon JDBC Driver for Apache Hive.

### Knowledge Prerequisites

To use the Amazon JDBC Driver for Apache Hive, the following knowledge is helpful:

- Familiarity with the platform on which you are using the Amazon JDBC Driver for Apache Hive
- Ability to use the data source to which the Amazon JDBC Driver for Apache Hive is connecting
- An understanding of the role of JDBC technologies in connecting to a data source
- 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.

Underline is not used.



The pencil icon indicates a short note appended to a paragraph.



The star icon indicates an important comment related to the preceding paragraph.



The thumbs up icon indicates a practical tip or suggestion.

# Table of Contents

Introduction .....	7
System Requirements .....	8
Amazon JDBC Driver for Apache Hive Files .....	9
Using the Amazon JDBC Driver for Apache Hive .....	10
Setting the Class Path .....	10
Initializing the Driver Class .....	10
Building the Connection URL .....	11
Java Sample Code .....	13
Configuring Authentication .....	17
Using No Authentication .....	17
Using Kerberos .....	17
Using User Name .....	18
Using User Name and Password .....	18
Configuring SSL .....	19
Features .....	20
SQL Query versus HiveQL Query .....	20
Data Types .....	20
Catalog and Schema Support .....	21
Contact Us .....	22
Appendix A Authentication Options .....	23
Using No Authentication .....	24
Using Kerberos .....	24
Using User Name .....	24
Using User Name and Password .....	24
Appendix B Configuring Kerberos Authentication for Windows .....	25
Downloading and Installing MIT Kerberos for Windows .....	25
Using the MIT Kerberos Ticket Manager to Get Tickets .....	25
Using the Driver to Get Tickets .....	26
Using an Existing Subject to Authenticate the Connection .....	28

Appendix C Driver Configuration Options .....	30
AllowSelfSignedCerts .....	30
AuthMech .....	30
CAIssuedCertNamesMismatch .....	30
CatalogSchemaSwitch .....	31
DecimalColumnScale .....	31
DefaultStringColumnLength .....	31
DelegationUID .....	32
KrbHostFQDN .....	32
KrbRealm .....	32
KrbServiceName .....	33
PreparedMetaLimitZero .....	33
PWD .....	33
RowsFetchedPerBlock .....	33
SocketTimeout .....	34
SSL .....	34
SSLKeyStore .....	34
SSLKeyStorePwd .....	35
SSLTrustStore .....	35
SSLTrustStorePwd .....	35
UID .....	36
UseNativeQuery .....	36
zk .....	36
Appendix D Kerberos Encryption Strength and the JCE Policy Files Extension ....	38

## Introduction

The Amazon JDBC Driver for Apache Hive is used for direct SQL and HiveQL access to Apache Hadoop / Hive distributions, enabling Business Intelligence (BI), analytics, and reporting on Hadoop / Hive-based data. The driver efficiently transforms an application's SQL query into the equivalent form in HiveQL, which is a subset of SQL-92. If an application is Hive-aware, then the driver is configurable to pass the query through to the database for processing. The driver interrogates Hive to obtain schema information to present to a SQL-based application. Queries, including joins, are translated from SQL to HiveQL. For more information about the differences between HiveQL and SQL, see [Features](#) on page 20.

The Amazon JDBC Driver for Apache Hive complies with the JDBC 3.0, 4.0 and 4.1 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.

This guide is suitable for users who want to access data residing within Hive 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 computer where you use the Amazon JDBC Driver for Apache Hive must have Java Runtime Environment (JRE) installed. The version of JRE that must be installed depends on the version of the JDBC API you are using with the driver. [Table 1](#) lists the required version of JRE for each version of the JDBC API.

**Table 1. Amazon JDBC Driver for Apache Hive System Requirements**

JDBC API Version	JRE Version
3.0	4.0 or 5.0
4.0	6.0 or later
4.1	7.0 or later

The Amazon JDBC Driver for Apache Hive supports Hive 0.11, 0.12, 0.13, 0.14, 1.0, and 1.1.



## Amazon JDBC Driver for Apache Hive Files

The Amazon JDBC Driver for Apache Hive is delivered in the following ZIP archives, where *version* is the version number of the driver:

- Amazon\_HiveJDBC3\_*version*.zip
- Amazon\_HiveJDBC4\_*version*.zip
- Amazon\_HiveJDBC41\_*version*.zip

Each archive contains the driver supporting the JDBC API version indicated in the archive name.

The archives contain the following file and folder structure, where *LibVersion* is the version number of the library and *APIVersion* is the JDBC API version that the driver supports:

- HiveJDBC*APIVersion*
  - hive\_metastore.jar
  - hive\_service.jar
  - HiveJDBC*APIVersion*.jar
  - libfb303-*LibVersion*.jar
  - libthrift-*LibVersion*.jar
  - log4j-*LibVersion*.jar
  - ql.jar
  - slf4j-api-*LibVersion*.jar
  - slf4j-log4j12-*LibVersion*.jar
  - TCLIServiceClient.jar
  - zookeeper-*LibVersion*.jar

## Using the Amazon JDBC Driver for Apache Hive

To access a Hive data warehouse using the Amazon JDBC Driver for Apache Hive, you need to set the following:

- Class path
- Driver class
- Connection URL

For sample code that demonstrates how to use the driver, see [Java Sample Code](#) on page 13.



The Amazon JDBC Driver for Apache Hive provides read-only access to Hive data.

### Setting the Class Path

To use the Amazon JDBC Driver for Apache Hive, you must set the class path to include all the JAR files from the ZIP archive containing the driver that you are using.

The class path is the path that the Java Runtime Environment searches for classes and other resource files. For more information, see the topic *Setting the Class Path* in the Java SE Documentation at

<http://docs.oracle.com/javase/7/docs/technotes/tools/windows/classpath.html>

### Initializing the Driver Class

Before connecting to the Hive server, initialize the appropriate class for the Hive server and your application.

The following is a list of the classes used to connect the Amazon JDBC Driver for Apache Hive to Hive Server 1 and Hive Server 2 instances. The Driver classes extend `java.sql.Driver`, and the DataSource classes extend `javax.sql.DataSource` and `javax.sql.ConnectionPoolDataSource`

To support JDBC 3.0, classes with the following fully-qualified class names (FQCNs) are available:

- `com.amazon.hive.jdbc3.HS1Driver`
- `com.amazon.hive.jdbc3.HS2Driver`
- `com.amazon.hive.jdbc3.HS1DataSource`
- `com.amazon.hive.jdbc3.HS2DataSource`

To support JDBC 4.0, classes with the following FQCNs are available:

- `com.amazon.hive.jdbc4.HS1Driver`
- `com.amazon.hive.jdbc4.HS2Driver`

- `com.amazon.hive.jdbc4.HS1DataSource`
- `com.amazon.hive.jdbc4.HS2DataSource`

To support JDBC 4.1, classes with the following FQCNs are available:

- `com.amazon.hive.jdbc41.HS1Driver`
- `com.amazon.hive.jdbc41.HS2Driver`
- `com.amazon.hive.jdbc41.HS1DataSource`
- `com.amazon.hive.jdbc41.HS2DataSource`

## Building the Connection URL

Use the connection URL to supply connection information to the data source that you are accessing. The connection URL for the Amazon JDBC Driver for Apache Hive takes the following form:

```
jdbc:Subprotocol://Host:Port  
[/Schema];Property1=Value;Property2=Value;...
```

The placeholders in the connection URL are defined as follows:

- *Subprotocol* is the value **hive** if you are connecting to a Hive Server 1 instance. If you are connecting to a Hive Server 2 instance, then use the value **hive2**.
- *Host* is the DNS or IP address of the server hosting the Hive data warehouse.
- *Port* is the port to connect to on *Host*.
- *Schema* is the name of the schema/database you want to access. Specifying a schema is optional. If you do not specify a schema, then the schema named **default** is used.



You can still issue queries on other schemas by explicitly specifying the schema in the query. To inspect your databases and determine the appropriate schema to use, type the **show databases** command at the Hive command prompt.

- *Property* is any one of the connection properties that you can specify. For information about the properties available in the driver, see [Driver Configuration Options](#) on page 30.



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

If a connection property key does not match any of the connection properties specified in [Driver Configuration Options](#) on page 30, then the driver will attempt to apply the property as a Hive server-side property for the client session.

For example, to connect to a Hive Server 2 instance installed on the local computer and authenticate the connection using a user name and password, you would use the following connection URL:

```
jdbc:hive  
2://localhost:10000;AuthMech=3;UID=UserName;PWD=Password
```

*UserName* and *Password* specify credentials for an existing user on the host running Hive Server 2.



If you use Hive Server 2 (**hive2**) and no parameters are specified, the UID will default to **hive** and AuthMech will default to **2**.

For more information about the properties that you can use in the connection URL, see [Driver Configuration Options](#) on page 30.

## Java Sample Code

The following Java code provides an example demonstrating how to use the JDBC API to do the following:

- Register the Amazon JDBC Driver for Apache Hive
- Establish a connection to a Hive database
- Query the database
- Parse a result set
- Handle exceptions
- Clean up to avoid memory leakage



To use the Amazon JDBC Driver for Apache Hive in an application, you must include all the JAR files from the ZIP archive in the class path for your Java project.

```
// java.sql packages are required
import java.sql.*;

class AmazonJDBCHiveExample {

    // Define a string as the fully qualified class name
    // (FQCN) of the desired JDBC driver
    static String JDBCdriver =
        "com.amazon.hive.jdbc3.HS1Driver";
    // Define a string as the connection URL
    private static final String CONNECTION_URL =
        "jdbc:hive://192.168.1.1:10000";

    public static void main(String[] args) {

        Connection con = null;
        Statement stmt = null;
        ResultSet rs = null;

        // Define a plain query
        String query = "SELECT first_name, last_name, emp_id
        FROM default.emp";
        // Define a parametrized query
```

```
String prepQuery = "SELECT first_name, last_name,  
emp_id FROM default.emp where store_id = ?";  
  
try {  
  
    // Register the driver using the class name  
    Class.forName(JDBC_DRIVER);  
  
    // Establish a connection using the connection  
    // URL  
    con = DriverManager.getConnection(CONNECTION_  
URL);  
  
    // Create a Statement object for sending SQL  
    // statements to the database  
    stmt = con.createStatement();  
  
    // Execute the SQL statement  
    rs = stmt.executeQuery(query);  
  
    // Display a header line for output appearing in  
    // the Console View  
    System.out.printf("%20s%20s%20s\r\n", "FIRST  
NAME", "LAST NAME" , "EMPLOYEE ID");  
  
    // Step through each row in the result set  
    // returned from the database  
    while(rs.next()) {  
        // Retrieve values from the row where the  
        // cursor is currently positioned using  
        // column names  
        String FirstName = rs.getString("first_  
name");  
        String LastName = rs.getString("last_name");  
        String EmployeeID = rs.getString("emp_id");
```

```
        // Display values in columns 20 characters
        // wide in the Console View using the
        // Formatter
        System.out.printf("%20s%20s%20s\r\n",
            FirstName, LastName, EmployeeID);
    }
    // Create a prepared statement
    PreparedStatement prep = m_conn.prepareStatement
        (prepQuery);

    // Bind the query parameter with a value
    prep.setInt(1, 204);
    // Execute the query
    rs = prep.execute();
    // Step through each row in the result set
    // returned from the database
    while(rs.next()) {
        // Retrieve values from the row where the
        // cursor is currently positioned using
        // column names
        String FirstName = rs.getString("first_
            name");
        String LastName = rs.getString("last_name");
        String EmployeeID = rs.getString("emp_id");

        // Display values in columns 20 characters
        // wide in the Console View using the
        // Formatter
        System.out.printf("%20s%20s%20s\r\n",
            FirstName, LastName, EmployeeID);
    }

} catch (SQLException se) {
    // Handle errors encountered during interaction
    // with the data source
} catch (Exception e) {
    // Handle other errors
```

```
    } finally {
        // Perform clean up
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException se1) {
            // Log this
        }

        try {
            if (stmt != null) {
                stmt.close();
            }
        } catch (SQLException se2) {
            // Log this
        }

        try {
            if (prep != null) {
                prep.close();
            }
        } catch (SQLException se3) {
            // Log this
        }

        try {
            if (con != null) {
                con.close();
            }
        } catch (SQLException se4) {
            // Log this
        } // End try
    } // End try
} // End main
} // End AmazonJDBCHiveExample
```



## Configuring Authentication


The Amazon JDBC Driver for Apache Hive supports the following authentication mechanisms:

- No Authentication
- Kerberos
- User Name
- User Name and Password


You configure the authentication mechanism that the driver uses to connect to Hive by specifying the relevant properties in the connection URL.

For information about selecting an appropriate authentication mechanism when using the Amazon JDBC Driver for Apache Hive, see [Authentication Options](#) on page 23.

For information about the properties you can use in the connection URL, see [Driver Configuration Options](#) on page 30.

 In addition to authentication, you can configure the driver to connect over SSL. For more information, see [Configuring SSL](#) on page 19.

### Using No Authentication

 When connecting to a Hive server of type Hive Server 1, you must use No Authentication.

#### To configure a connection without authentication:


- Set the AuthMech property to 0

For example:

```
jdbc:hive2://localhost:10000;AuthMech=0
```

### Using Kerberos

Kerberos must be installed and configured before you can use this authentication mechanism. For information about configuring and operating Kerberos on Windows, see [Configuring Kerberos Authentication for Windows](#) on page 25. For other operating systems, refer to the MIT Kerberos documentation.

 This authentication mechanism is available only for Hive Server 2.

#### To configure Kerberos authentication:

1. Set the AuthMech property to 1

2. If your Kerberos setup does not define a default realm or if the realm of your Hive server is not the default, then set the KrbRealm property to the realm of the Hive server.

OR

To use the default realm defined in your Kerberos setup, do not set the KrbRealm property.

3. Set the KrbHostFQDN property to the fully qualified domain name of the Hive server host.
4. Set the KrbServiceName property to the service name of the Hive server.

For example:

```
jdbc:  
hive2://localhost:10000;AuthMech=1;KrbRealm=EXAMPLE.COM;  
KrbHostFQDN=hs2.example.com;KrbServiceName=hive
```

## Using User Name

This authentication mechanism requires a user name but does not require a password. The user name labels the session, facilitating database tracking.



This authentication mechanism is available only for Hive Server 2. Most default configurations of Hive Server 2 require User Name authentication.

### To configure User Name authentication:

1. Set the AuthMech property to 2
2. Set the UID property to an appropriate user name for accessing the Hive server.

For example:

```
jdbc:hive2://localhost:10000;AuthMech=2;UID=hs2
```

## Using User Name and Password

This authentication mechanism requires a user name and a password.



This authentication mechanism is available only for Hive Server 2.

### To configure User Name and Password authentication:

1. Set the AuthMech property to 3
2. Set the UID property to an appropriate user name for accessing the Hive server.
3. Set the PWD property to the password corresponding to the user name you provided in step 2.

For example:

```
jdbc:hive2://localhost:10000;AuthMech=3;UID=hs2;PWD=*****
```

## Configuring SSL

If you are connecting to a Hive server that has Secure Sockets Layer (SSL) enabled, then you can configure the driver to connect to an SSL-enabled socket.

SSL connections require a KeyStore and a TrustStore. You can create a TrustStore and configure the driver to use it, or allow the driver to use one of the default TrustStores. If you do not configure the driver to use a specific TrustStore, then the driver uses the Java TrustStore `jssecacerts`. If `jssecacerts` is not available, then the driver uses `cacerts` instead.

### To configure SSL:

1. To create a KeyStore and configure the driver to use it, do the following:
  - a) Create a KeyStore containing your signed, trusted SSL certificate.
  - b) Set the `SSLKeyStore` property to the full path of the KeyStore, including the file name.
  - c) Set the `SSLKeyStorePwd` property to the password for the KeyStore.
2. Optionally, to create a TrustStore and configure the driver to use it, do the following:
  - a) Create a TrustStore containing your signed, trusted SSL certificate.
  - b) Set the `SSLTrustStore` property to the full path of the TrustStore, including the file name.
  - c) Set the `SSLTrustStorePwd` property to the password for the TrustStore.
3. Set the `SSL` property to 1
4. Optionally, to allow the SSL certificate used by the server to be self-signed, set the `AllowSelfSignedCerts` property to 1
5. Optionally, to allow the common name of a CA-issued certificate to not match the host name of the Hive server, set the `CAIssuedCertNamesMismatch` property to 1



For self-signed certificates, the driver always allows the common name of the certificate to not match the host name.

For example:

```
jdbc:hive2://localhost:10000;AuthMech=3;SSL=1;  
SSLKeyStore=C:\\Users\\bsmith\\Desktop\\keystore.jks;  
SSLKeyStorePwd=*****;UID=hs2;PWD=*****
```



For more information about the connection properties used in SSL connections, see [Driver Configuration Options](#) on page 30

## Features

More information is provided on the following features of the Amazon JDBC Driver for Apache Hive:

- [SQL Query versus HiveQL Query](#) on page 20
- [Data Types](#) on page 20
- [Catalog and Schema Support](#) on page 21

## SQL Query versus HiveQL Query

The native query language supported by Hive is HiveQL. HiveQL is a subset of SQL-92. However, the syntax is different enough that most applications do not work with native HiveQL.

## Data Types

The Amazon JDBC Driver for Apache Hive supports many common data formats, converting between Hive, SQL, and Java data types.

[Table 2](#) lists the supported data type mappings.

**Table 2. Supported Data Types**

Hive Type	SQL Type	Java Type
BIGINT	BIGINT	java.math.BigInteger
BINARY	VARBINARY	byte[]
BOOLEAN	BOOLEAN	Boolean
CHAR (Available only in Hive 0.13.0 or later)	CHAR	String
DATE	DATE	java.sql.Date
DECIMAL (In Hive 0.13 and later, you can specify scale and precision when creating tables using the DECIMAL data type.)	DECIMAL	java.math.BigDecimal

Hive Type	SQL Type	Java Type
DOUBLE	DOUBLE	Double
INT	INTEGER	Long
FLOAT	REAL	Float
SMALLINT	SMALLINT	Integer
TINYINT	TINYINT	Short
TIMESTAMP	TIMESTAMP	java.sql.Timestamp
VARCHAR (Available only in Hive 0.12.0 or later)	VARCHAR	String

The aggregate types (ARRAY, MAP, STRUCT, and UNIONTYPE) are not yet supported. Columns of aggregate types are treated as VARCHAR columns in SQL and STRING columns in Java.

## Catalog and Schema Support

The Amazon JDBC Driver for Apache Hive supports both catalogs and schemas in order to make it easy for the driver to work with various JDBC applications. Since Hive only organizes tables into schemas/databases, the driver provides a synthetic catalog called “HIVE” under which all of the schemas/databases are organized. The driver also maps the JDBC schema to the Hive schema/database.



Setting the CatalogSchemaSwitch connection property to 1 will cause Hive catalogs to be treated as schemas in the driver as a restriction for filtering.

## Contact Us

For support, check the EMR Forum at <https://forums.aws.amazon.com/forum.jspa?forumID=52> or open a support case using the AWS Support Center at <https://aws.amazon.com/support>

## Appendix A Authentication Options

Hive Server 1 does not support authentication. You must configure the driver to use No Authentication.

Hive Server 2 supports the following authentication mechanisms:

- No Authentication
- Kerberos
- User Name
- User Name and Password

Most default configurations of Hive Server 2 require User Name authentication. If you are unable to connect to your Hive server using User Name authentication, then verify the authentication mechanism configured for your Hive server by examining the `hive-site.xml` file. Examine the following properties to determine which authentication mechanism your server is set to use:

- **hive.server2.authentication**—This property sets the authentication mode for Hive Server 2. The following values are available:
  - **NOSASL** disables the Simple Authentication and Security Layer (SASL).
  - **KERBEROS** enables Kerberos authentication.
  - **NONE** enables plain SASL transport. **NONE** is the default value.
  - **PLANSASL** enables user name and password authentication using a cleartext password mechanism.
- **hive.server2.enable.doAs**—If this property is set to the default value of **TRUE**, then Hive processes queries as the user who submitted the query. If this property is set to **FALSE**, then queries are run as the user that runs the `hiveserver2` process.

[Table 3](#) lists authentication mechanisms to configure for the driver based on the settings in the `hive-site.xml` file.

**Table 3. Hive Authentication Mechanism Configurations**

<b>hive.server2.authentication</b>	<b>hive.server2.enable.doAs</b>	<b>Driver Authentication Mechanism</b>
NOSASL	FALSE	No Authentication
KERBEROS	TRUE or FALSE	Kerberos
NONE	TRUE or FALSE	User Name
LDAP	TRUE or FALSE	User Name and Password



It is an error to set `hive.server2.authentication` to `NOSASL` and `hive.server2.enable.doAs` to `true`. This configuration will not prevent the service from starting up, but results in an unusable service.

For more information about authentication mechanisms, refer to the documentation for your Hadoop / Hive distribution. See also *Running Hadoop in Secure Mode* at [http://hadoop.apache.org/docs/r0.23.7/hadoop-project-dist/hadoop-common/ClusterSetup.html#Running\\_Hadoop\\_in\\_Secure\\_Mode](http://hadoop.apache.org/docs/r0.23.7/hadoop-project-dist/hadoop-common/ClusterSetup.html#Running_Hadoop_in_Secure_Mode)

## Using No Authentication

When `hive.server2.authentication` is set to `NOSASL`, you must configure your connection to use No Authentication.

## Using Kerberos

When connecting to a Hive server of type Hive Server 2 and `hive.server2.authentication` is set to `KERBEROS`, you must configure your connection to use Kerberos authentication.

## Using User Name

When connecting to a Hive server of type Hive Server 2 and `hive.server2.authentication` is set to `NONE`, you must configure your connection to use User Name authentication.

Validation of the credentials that you include depends on `hive.server2.enable.doAs`:

- If `hive.server2.enable.doAs` is set to `TRUE`, then the user name in the driver configuration must be an existing operating system user on the host that is running Hive Server 2.
- If `hive.server2.enable.doAs` is set to `FALSE`, then the user name in the driver configuration is ignored.

If no user name is specified in the driver configuration, then the driver defaults to using "hive" as the user name.

## Using User Name and Password

When connecting to a Hive server of type Hive Server 2 and the server is configured to use the SASL-PLAIN authentication mechanism with a user name and a password, you must configure your connection to use User Name and Password authentication.



## Appendix B Configuring Kerberos Authentication for Windows

You can configure your Kerberos setup so that you use the MIT Kerberos Ticket Manager to get the Ticket Granting Ticket (TGT), or configure the setup so that you can use the driver to get the ticket directly from the Key Distribution Center (KDC). Also, if a client application obtains a Subject with a TGT, it is possible to use that Subject to authenticate the connection.

### Downloading and Installing MIT Kerberos for Windows

#### To download and install MIT Kerberos for Windows:

1. To download the Kerberos installer for 64-bit computers, use the following download link from the MIT Kerberos website: <http://web.mit.edu/kerberos/dist/kfw/4.0/kfw-4.0.1-amd64.msi>

The 64-bit installer includes both 32-bit and 64-bit libraries.

OR

To download the Kerberos installer for 32-bit computers, use the following download link from the MIT Kerberos website: <http://web.mit.edu/kerberos/dist/kfw/4.0/kfw-4.0.1-i386.msi>

The 32-bit installer includes 32-bit libraries only.


2. To run the installer, double-click the .msi file that you downloaded in step 1.
3. Follow the instructions in the installer to complete the installation process.
4. When the installation completes, click **Finish**

### Using the MIT Kerberos Ticket Manager to Get Tickets

#### Setting the KRB5CCNAME Environment Variable

You must set the KRB5CCNAME environment variable to your credential cache file.


#### To set the KRB5CCNAME environment variable:

1. Click the **Start** button , then right-click **Computer**, and then click **Properties**
2. Click **Advanced System Settings**
3. In the System Properties dialog box, click the **Advanced** tab, and then click **Environment Variables**
4. In the Environment Variables dialog box, under the System variables list, click **New**
5. In the New System Variable dialog box, in the Variable name field, type **KRB5CCNAME**

6. In the Variable value field, type the path for your credential cache file. For example, type **C:\KerberosTickets.txt**
7. Click **OK** to save the new variable.
8. Ensure that the variable appears in the System variables list.
9. Click **OK** to close the Environment Variables dialog box, and then click **OK** to close the System Properties dialog box.
10. To ensure that Kerberos uses the new settings, restart your computer.

## Getting a Kerberos Ticket

### To get a Kerberos ticket:

1. Click the **Start** button , then click **All Programs**, and then click the **Kerberos for Windows (64-bit)** or **Kerberos for Windows (32-bit)** program group.
2. Click **MIT Kerberos Ticket Manager**
3. In the MIT Kerberos Ticket Manager, click **Get Ticket**
4. In the Get Ticket dialog box, type your principal name and password, and then click **OK**

If the authentication succeeds, then your ticket information appears in the MIT Kerberos Ticket Manager.

## Authenticating to the Hive Server

### To authenticate to the Hive server:

- Use a connection string that has the following properties defined:
  - AuthMech
  - KrbHostFQDN
  - KrbRealm
  - KrbServiceName

For detailed information about these properties, see [Driver Configuration Options](#) on page 30.

## Using the Driver to Get Tickets

### Deleting the KRB5CCNAME Environment Variable

To enable the driver to get Ticket Granting Tickets (TGTs) directly, you must ensure that the KRB5CCNAME environment variable has not been set.

### To delete the KRB5CCNAME environment variable:

1. Click the **Start** button , then right-click **Computer**, and then click **Properties**

2. Click **Advanced System Settings**
3. In the System Properties dialog box, click the **Advanced** tab and then click **Environment Variables**
4. In the Environment Variables dialog box, check if the KRB5CCNAME variable appears in the System variables list. If the variable appears in the list, then select the variable and click **Delete**
5. Click **OK** to close the Environment Variables dialog box, and then click **OK** to close the System Properties dialog box.

## Setting Up the Kerberos Configuration File

### To set up the Kerberos configuration file:

1. Create a standard krb5.ini file and place it in the C:\Windows directory.
2. Ensure that the KDC and Admin server specified in the krb5.ini file can be resolved from your terminal. If necessary, modify "C:\Windows\System32\drivers\etc\hosts".

## Setting Up the JAAS Login Configuration File

### To set up the JAAS login configuration file:

1. Create a JAAS login configuration file that specifies a keytab file and "doNotPrompt=true"

For example:

```
Client {  
    com.sun.security.auth.module.Krb5LoginModule  
    required  
        useKeyTab=true  
        keyTab="PathToTheKeyTab"  
        principal="amazon@AMAZON"  
        doNotPrompt=true;  
};
```

2. Set the java.security.auth.login.config environment variable to the location of the JAAS file.

For example: C:\KerberosLoginConfig.ini

## Authenticating to the Hive Server

### To authenticate to the Hive server:

- Use a connection string that has the following properties defined:
  - AuthMech
  - KrbHostFQDN

- KrbRealm
- KrbServiceName

For detailed information about these properties, see [Driver Configuration Options](#) on page 30.

## Using an Existing Subject to Authenticate the Connection

If the client application obtains a Subject with a TGT, then that Subject can be used to authenticate the connection to the server.

### To use an existing Subject to authenticate the connection:

1. Create a PrivilegedAction for establishing the connection to the database.

For example:

```
// Contains logic to be executed as a privileged action
public class AuthenticateDriverAction
implements PrivilegedAction<Void>
{
    // The connection, which is established as a
    // PrivilegedAction
    Connection con;

    // Define a string as the connection URL
    static String ConnectionURL =
        "jdbc:hive2://192.168.1.1:10000";

    /**
     * Logic executed in this method will have access to
     * the
     * Subject that is used to "doAs". The driver will
     * get
     * the Subject and use it for establishing a
     * connection
     * with the server.
     */
    @Override
    public Void run()
    {
        try
```

```
        {  
            // Establish a connection using the  
            connection URL  
            con = DriverManager.getConnection  
            (ConnectionURL);  
        }  
        catch (SQLException e)  
        {  
            // Handle errors that are encountered during  
            // interaction with the data source  
            e.printStackTrace();  
        }  
        catch (Exception e)  
        {  
            // Handle other errors  
            e.printStackTrace();  
        }  
        return null;  
    }  
}
```

2. Run the `PrivilegedAction` using the existing `Subject`, and then use the connection.

For example:

```
// Create the action  
AuthenticateDriverAction authenticateAction = new  
AuthenticateDriverAction();  
// Establish the connection using the Subject for  
// authentication.  
Subject.doAs(loginConfig.getSubject(),  
authenticateAction);  
// Use the established connection.  
authenticateAction.con;
```

## Appendix C Driver Configuration Options

[Appendix C](#) lists and describes the properties that you can use to configure the behavior of the Amazon JDBC Driver for Apache Hive.



You can set configuration properties using the connection URL. For more information, see [Building the Connection URL](#) on page 11.

### AllowSelfSignedCerts

Default Value	Required
0	No

#### Description

When this property is set to 0, the SSL certificate used by the server cannot be self-signed.

When this property is set to 1, the SSL certificate used by the server can be self-signed.



This property is applicable only when SSL connections are enabled.

### AuthMech

Default Value	Required
0	No

#### Description

The authentication mechanism to use. Set the value to one of the following numbers:

- **0** for No Authentication
- **1** for Kerberos
- **2** for User Name
- **3** for User Name and Password


### CAIssuedCertNamesMismatch

Default Value	Required
0	No

## Description

When this property is set to 0, the name of the CA-issued SSL certificate must match the host name of the Hive server.

When this property is set to 1, the names of the certificate and the host name of the server are allowed to mismatch.

 This property is applicable only when SSL connections are enabled.

## CatalogSchemaSwitch

Default Value	Required
0	No

## Description

When this property is set to 1, the driver treats Hive catalogs as schemas as a restriction for filtering.

When this property is set to 0, Hive catalogs are treated as catalogs, and Hive schemas are treated as schemas.

## DecimalColumnScale

Default Value	Required
10	No

## Description

The maximum number of digits to the right of the decimal point for numeric data types.

## DefaultStringColumnLength

Default Value	Required
255	No

## Description

The maximum data length for STRING columns. The range of DefaultStringColumnLength is 0 to 32,767.

By default, the columns metadata for Hive does not specify a maximum data length for STRING columns.

## DelegationUID

Default Value	Required
None	No

### Description

Use this option to delegate all operations against Hive to a user that is different than the authenticated user for the connection.



This option is applicable only when connecting to a Hive Server 2 instance that supports this feature.

## KrbHostFQDN

Default Value	Required
None	Yes, if AuthMech=1 (Kerberos)

### Description

The fully qualified domain name of the Hive Server 2 host.

## KrbRealm

Default Value	Required
Depends on Kerberos configuration.	No

### Description

The realm of the Hive Server 2 host.

If your Kerberos configuration already defines the realm of the Hive Server 2 host as the default realm, then you do not need to configure this option.



## KrbServiceName

Default Value	Required
None	Yes, if AuthMech=1 (Kerberos)

## Description

The Kerberos service principal name of the Hive server.

## PreparedMetaLimitZero

Default Value	Required
0	No

## Description

When this property is set to 1, the `PreparedStatement.getMetadata()` call will request metadata from the server with "LIMIT 0".

## PWD

Default Value	Required
None	Yes, if AuthMech=3 (User Name and Password)

## Description

The password corresponding to the user name that you provided using the property [UID](#) on page [36](#).

## RowsFetchedPerBlock

Default Value	Required
10000	No

## Description

The maximum number of rows that a query returns at a time.

Any positive 32-bit integer is a valid value, but testing has shown that performance gains are marginal beyond the default value of 10000 rows.

## SocketTimeout

Default Value	Required
0	No

### Description

The number of seconds after which Hive closes the connection with the client application if the connection is idle. The default value of 0 indicates that an idle connection is not closed.


## SSL

Default Value	Required
0	No

### Description

When this property is set to 1, the driver communicates with the Hive server through an SSL-enabled socket.

When this property is set to 0, the driver does not connect to SSL-enabled sockets.

 SSL is configured independently of authentication. When authentication and SSL are both enabled, the driver performs the specified authentication method over an SSL connection.

## SSLKeyStore

Default Value	Required
None	Yes, if SSL=1

### Description

The full path and file name of the Java KeyStore containing an SSL certificate to use during authentication.

See also the property [SSLKeyStorePwd](#) on page 35.

## SSLKeyStorePwd

Default Value	Required
None	Yes, if SSL=1

### Description

The password for accessing the Java KeyStore that you specified using the property [SSLKeyStore](#) on page 34.

## SSLTrustStore

Default Value	Required
jssecacerts, if it exists.  If jssecacerts does not exist, then cacerts is used. The default location of cacerts is jre\lib\security\	No

### Description

The full path and file name of the Java TrustStore containing an SSL certificate to use during authentication.

See also the property [SSLTrustStorePwd](#) on page 35.

## SSLTrustStorePwd

Default Value	Required
None	Yes, if using a TrustStore.

### Description

The password for accessing the Java TrustStore that you specified using the property [SSLTrustStore](#) on page 35.

## UID

Default Value	Required
hive	Yes, if AuthMech=3 (User Name and Password) No, if AuthMech=2 (User Name)

## Description

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


## UseNativeQuery

Default Value	Required
0	No

## Description

When this option is enabled (1), the driver does not transform the queries emitted by an application, so the native query is used.

When this option is disabled (0), the driver transforms the queries emitted by an application and converts them into an equivalent form in HiveQL.

 If the application is Hive-aware and already emits HiveQL, then enable this option to avoid the extra overhead of query transformation.

## zk

Default Value	Required
None	No

## Description

The connection string to one or more ZooKeeper quorums, written in the following format:

*ZK\_IP:ZK\_Port/ZK\_Namespace*

For example:

`jdbc:hive2://zk=192.168.0.1:2181/hiveserver2`

Use this option to enable the Dynamic Service Discovery feature, which allows you to connect to Hive servers that are registered against a ZooKeeper service by connecting to the ZooKeeper service.

You can specify multiple quorums in a comma-separated list. If connection to a quorum fails, the driver will attempt to connect to the next quorum in the list.

## Appendix D Kerberos Encryption Strength and the JCE Policy Files Extension

If the encryption being used in your Kerberos environment is too strong, you may encounter the error message "Unable to connect to server: GSS initiate failed" when trying to use the driver to connect to a Kerberos-enabled cluster. Typically, Java vendors only allow encryption strength up to 128 bits by default. If you are using greater encryption strength in your environment (for example, 256-bit encryption), then you may encounter this error.

### Diagnosing the Issue

If you encounter the error message "Unable to connect to server: GSS initiate failed", confirm that it is occurring due to encryption strength by enabling Kerberos layer logging in the JVM and then checking if the log output contains the error message "KrbException: Illegal key size".

#### To enable Kerberos layer logging in a Sun JVM:

- In the Java command you use to start the application, pass in the following argument:

```
-Dsun.security.krb5.debug=true
```

OR

Add the following code to the source code of your application:

```
System.setProperty("sun.security.krb5.debug", "true")
```

#### To enable Kerberos layer logging in an IBM JVM:

- In the Java command you use to start the application, pass in the following arguments:

```
-Dcom.ibm.security.krb5.Krb5Debug=all
```

```
-Dcom.ibm.security.jgss.debug=all
```

OR

Add the following code to the source code of your application:

```
System.setProperty  
("com.ibm.security.krb5.Krb5Debug", "all");  
System.setProperty("com.ibm.security.jgss.debug", "all");
```

### Resolving the Issue

After you confirm that the error is occurring due to encryption strength, you can resolve the issue by downloading and installing the *Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files* extension from your Java vendor. Refer to the instructions from the vendor to install the files to the correct location.



Consult your company's policy to ensure that you are allowed to enable encryption strengths in your environment that are greater than what the JVM allows by default.

If the issue is not resolved after you install the JCE policy files extension, then restart your machine and try your connection again. If the issue persists even after you restart your machine, then verify which directories the JVM is searching to find the JCE policy files extension. To print out the search paths that your JVM currently uses to find the JCE policy files extension, modify your Java source code to print the return value of the following call:

```
System.getProperty("java.ext.dirs")
```