Amazon JDBC Connector for Apache Impala

Installation and Configuration Guide

Version 2.6.27

March 2022

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

About This Guide

Purpose

The *Amazon JDBC Connector for Apache Impala Installation and Configuration Guide* explains how to install and configure the Amazon JDBC Connector for Apache Impala on all supported platforms. The guide also provides details related to features of the connector.

Audience

The guide is intended for end users of the Amazon JDBC Connector for Apache Impala.

Knowledge Prerequisites

To use the Amazon JDBC Connector for Apache Impala, the following knowledge is helpful:

- Familiarity with the platform on which you are using the Amazon JDBC Connector for Apache Impala
- Ability to use the data store to which the Amazon JDBC Connector for Apache Impala 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.

Contents

About the Amazon JDBC Connector for Apache Impala

The Amazon JDBC Connector for Apache Impala is used for direct SQL and Impala SQL access to Apache Hadoop / Impala distributions, enabling Business Intelligence (BI), analytics, and reporting on Hadoop / Impala-based data. The connector efficiently transforms an application's SQL query into the equivalent form in Impala SQL, which is a subset of SQL-92. If an application is Impala-aware, then the connector is configurable to pass the query through to the database for processing. The connector interrogates Impala to obtain schema information to present to a SQL-based application. Queries, including joins, are translated from SQL to Impala SQL. For more information about the differences between Impala SQL and SQL, see Features on page 33.

The Amazon JDBC Connector for Apache Impala 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 connector, which connects an application to the database.

This guide is suitable for users who want to access data residing within Impala from their desktop environment. Application developers might also find the information helpful. Refer to your application for details on connecting via JDBC.

System Requirements

Each machine where you use the Amazon JDBC Connector for Apache Impala must have Java Runtime Environment (JRE) 7.0 or 8.0 or 11.0 installed. If you are using the connector with JDBC API version 4.2, then you must use JRE 8.0 or 11.0.

The connector supports Apache Impala versions 2.8 through 3.4.

Amazon JDBC Connector for Apache Impala Files

The Amazon JDBC Connector for Apache Impala is delivered in the following ZIP archives, where *[Version]* is the version number of the connector:

- `Amazon_ImpalaJDBC41_[Version].zip`
- `Amazon_ImpalaJDBC42_[Version].zip`

The archive contains the connector supporting the JDBC API version indicated in the archive name, as well as release notes and third-party license information. In addition, the required third-party libraries and dependencies are packaged and shared in the connector JAR file in the archive.

Installing and Using the Amazon JDBC Connector for Apache Impala

To install the Amazon JDBC Connector for Apache Impala on your machine, extract the files from the appropriate ZIP archive to the directory of your choice.

To access an Impala data store using the Amazon JDBC Connector for Apache Impala, you need to configure the following:

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

> ⚠️ **Important:**
>
> The Amazon JDBC Connector for Apache Impala provides read-only access to Impala data.

Referencing the JDBC Connector Libraries

Before you use the Amazon JDBC Connector for Apache Impala, the JDBC application or Java code that you are using to connect to your data must be able to access the connector JAR files. In the application or code, specify all the JAR files that you extracted from the ZIP archive.

Using the Connector in a JDBC Application

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

Using the Connector in Java Code

You must include all the connector 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/unix/classpath.html

Registering the Connector Class

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

The following classes are used to connect the Amazon JDBC Connector for Apache Impala to Impala data stores:

- The `Driver` classes extend `java.sql.Driver`.
- The `DataSource` classes extend `javax.sql.DataSource` and `javax.sql.ConnectionPoolDataSource`.

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

- `com.amazon.impala.jdbc.Driver`
- `com.amazon.impala.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;
    Class.forName(DRIVER_CLASS);
    connection = DriverManager.getConnection(CONNECTION_
    URL);
    return connection;

}
```

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

```
private static Connection connectViaDS() throws Exception
{

    Connection connection = null;
    Class.forName(DRIVER_CLASS);
    DataSource ds = new com.amazon.impala.jdbc.DataSource
    ();
    ds.setURL(CONNECTION_URL);
    connection = ds.getConnection();
    return connection;

}
```

Building the Connection URL

Use the connection URL to supply connection information to the data store that you are accessing. The following is the format of the connection URL for the Amazon JDBC Connector for Apache Impala, where *[Host]* is the DNS or IP address of the Impala server and *[Port]* is the number of the TCP port that the server uses to listen for client requests:

```
jdbc:impala://[Host]:[Port]
```

> **ⓘ Note:**
>
> By default, the connector uses port 28000 when `TransportMode` is set to `http`, and 21050 when `TransportMode` is not set or is set to `sasl` or `binary`.

By default, the connector uses the schema named **default**.

You can specify optional settings such as the schema to use or any of the connection properties supported by the connector. For a list of the properties available in the connector, see Connector Configuration Options on page 37.

> **ⓘ Note:**
>
> If you specify a property that is not supported by the connector, then the connector attempts to apply the property as a Impala server-side property for the client session. For more information, see Configuring Server-Side Properties on page 29.

The following is the format of a connection URL that specifies some optional settings:

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

For example, to connect to port 18000 on an Impala server installed on the local machine, use a schema named default2, and authenticate the connection using a user name and password, you would use the following connection URL:

```
jdbc:impala://node1.example.com:18000/default2;AuthMech=3;
UID=amazon;PWD=amazon
```

> **⚠ Important:**
>
> - Properties are case-sensitive.
> - Do not duplicate properties in the connection URL.

Configuring Authentication

The Amazon JDBC Connector for Apache Impala supports the following authentication mechanisms:

- No Authentication
- Kerberos
- User Name
- User Name And Password
- Single Sign-On (SSO)

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

For information about configuring the authentication mechanism that Impala uses, see the Impala documentation:
http://www.cloudera.com/content/cloudera/en/documentation.html.

For information about the properties you can use in the connection URL, see Connector Configuration Options on page 37.

> **ⓘ Note:**
>
> In addition to authentication, you can configure the connector to connect over SSL. For more information, see Configuring SSL on page 27.

Using No Authentication

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 13.

To configure a connection without authentication:

➢ Set the `AuthMech` property to `0`.

For example:

```
jdbc:impala://localhost:21050;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 19. For other operating systems, see the MIT Kerberos documentation: http://web.mit.edu/kerberos/krb5-latest/doc/.

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 13.

> **ⓘ Note:**
>
> The connector also supports Kerberos constrained delegation. For more details on this, see Using Kerberos Constrained Delegation on page 25.

To configure default Kerberos authentication:

1. Set the `AuthMech` property to `1`.
2. To use the default realm defined in your Kerberos setup, do not set the `KrbRealm` property.

   If your Kerberos setup does not define a default realm or if the realm of your Impala server is not the default, then set the `KrbRealm` property to the realm of the Impala server.
3. Set the `KrbHostFQDN` property to the fully qualified domain name of the Impala server host.
4. If you are using Kerberos Constrained Delegation, set the `userGSSCredential` property to your Kerberos GSS Credential.
5. Optionally, specify how the connector obtains the Kerberos Subject by setting the `KrbAuthType` property as follows:
   - To configure the connector to automatically detect which method to use for obtaining the Subject, set the `KrbAuthType` property to `0`. Alternatively, do not set the `KrbAuthType` property.
   - Or, to create a LoginContext from a JAAS configuration and then use the Subject associated with it, set the `KrbAuthType` property to `1`.
   - Or, to create a LoginContext from a Kerberos ticket cache and then use the Subject associated with it, set the `KrbAuthType` property to `2`.

   For more detailed information about how the connector obtains Kerberos Subjects based on these settings, see KrbAuthType on page 42.

For example, the following connection URL connects to a Impala server with Kerberos enabled, but without SSL enabled:

```
jdbc:impala://node1.example.com:21050;AuthMech=1;
KrbRealm=EXAMPLE.COM;KrbHostFQDN=node1.example.com;
KrbServiceName=impala
```

In this example, Kerberos is enabled for JDBC connections, the Kerberos service principal name is impala/node1.example.com@EXAMPLE.COM, the host name for the data source is node1.example.com, and the server is listening on port 21050 for JDBC connections.

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.

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 13.

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 Impala server.

For example:

```
jdbc:impala://node1.example.com:21050;AuthMech=2;UID=impala
```

Using User Name And Password (LDAP)

This authentication mechanism requires a user name and a password. It is most commonly used with LDAP authentication.

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 13.

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 Impala server.

3. Set the `PWD` property to the password corresponding to the user name you provided.

For example, the following connection URL connects to a Impala server with LDAP authentication enabled:

```
jdbc:impala://node1.example.com:21050;AuthMech=3;
UID=impala;PWD=amazon;
```

In this example, user name and password (LDAP) authentication is enabled for JDBC connections, the LDAP user name is impala, the password is amazon, and the server is listening on port 21050 for JDBC connections.

Using Single Sign-On

Single Sign-On (SSO) is a process that allows network users to access all authorized network resources without having to log in to each resource separately. For example, implementing SSO for users within an organization allows each user to authenticate to Impala without providing a separate set of Impala credentials.

You specify the properties in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 13.

> ⚠ **Important:**
>
> SSL is required for this authentication method. For more information, see Configuring SSL on page 27.

To configure Single Sign-On authentication:

1. Set the `AuthMech` property to `12`.
2. Set the `TransportMode` property to `http`.
3. Optionally, set the `SSOWebServerTimeout` property to the number of seconds that the connector waits before timing out while waiting for a browser response.

For example:

```
jdbc:impala://node1.example.com:28000;AuthMech=12;
SSL=1;TransportMode=http;httpPath=cliservice;SSOWebServerTime
out=60;
```

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 connector 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 4.0.1:

1. Download the appropriate Kerberos installer:
   - For a 64-bit machine, 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.
   - For a 32-bit machine, 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.

   ⓘ Note:

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

2. To run the installer, double-click the `.msi` file that you downloaded.
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 **Start** 🌐, then right-click **Computer**, and then click **Properties**.
2. Click **Advanced System Settings**.
3. In the System Properties dialog box, on the **Advanced** tab, 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. Make sure 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. Restart your machine.

Getting a Kerberos Ticket

To get a Kerberos ticket:

1. Click **Start** 🌐, 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 Impala Server

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 13.

To authenticate to the Impala server:

➢ Use a connection URL that has the following properties defined:

- `AuthMech`
- `KrbHostFQDN`
- `KrbRealm`
- `KrbServiceName`

For detailed information about these properties, see Connector Configuration Options on page 37

Using the Connector to Get Tickets

Deleting the KRB5CCNAME Environment Variable

To enable the connector to get Ticket Granting Tickets (TGTs) directly, make sure 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. Make sure 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` system property to the location of the JAAS file.

   For example: `C:\KerberosLoginConfig.ini`.

Authenticating to the Impala Server

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see Building the Connection URL on page 13.

To authenticate to the Impala server:

➤ Use a connection URL that has the following properties defined:

- `AuthMech`
- `KrbHostFQDN`
- `KrbRealm`
- `KrbServiceName`

For detailed information about these properties, see Connector Configuration Options on page 37.

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:impala://192.168.1.1:21050";
```

```
/**
* Logic executed in this method will have access to the
* Subject that is used to "doAs". The connector 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 store
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;
```

Kerberos Encryption Strength and the JCE Policy Files Extension

If the encryption being used in your Kerberos environment is too strong, you might encounter the error message "Unable to connect to server: GSS initiate failed" when trying to use the connector 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 might 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:

➢ Choose one:

- 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:

➢ Choose one:

- 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.

> ⚠️ **Important:**
>
> Consult your company's policy to make sure 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")
```

Using Kerberos Constrained Delegation

The connector can also be configured to use Kerberos Constrained Delegation. This feature allows a service to obtain service tickets to a restricted list of other services running on specific servers on the network after it has been presented with a service ticket. For more details on the process see: https://technet.microsoft.com/en-ca/library/cc995228.aspx.

The `userGSSCredential` connection property can be used in the connection URL to pass in a GSSCredential object. The following sample code shows how to use the property to pass the GSSCredential into the connector using JDBC 4.1.

```
GSSCredential userCredential = [GSSCredential]
```

```
Driver driver = (Driver) Class.forName
("com.amazon.impala.jdbc.Driver").newInstance();
Properties properties = new Properties();
properties.put("userGSSCredential", userCredential);
Connection conn = driver.connect(

    "jdbc:impala
    ://node1.example.com:21050;AuthMech=1;KrbRealm=EXAMPL
    E.COM;
    KrbHostFQDN=node1.example.com;KrbServiceName=impala"
    ,properties);
```

Configuring SSL

> ℹ **Note:**
>
> In this documentation, "SSL" indicates both TLS (Transport Layer Security) and SSL (Secure Sockets Layer). The connector supports industry-standard versions of TLS/SSL.

If you are connecting to an Impala server that has Secure Sockets Layer (SSL) enabled, you can configure the connector to connect to an SSL-enabled socket. When connecting to a server over SSL, the connector uses one-way authentication to verify the identity of the server.

One-way authentication requires a signed, trusted SSL certificate for verifying the identity of the server. You can configure the connector to access a specific TrustStore or KeyStore that contains the appropriate certificate. If you do not specify a TrustStore or KeyStore, then the connector uses the default Java TrustStore named `jssecacerts`. If `jssecacerts` is not available, then the connector uses `cacerts` instead.

You provide this information to the connector in the connection URL. For more information about the syntax of the connection URL, see .

To configure SSL:

1. Set the `SSL` property to `1`.
2. If you are not using one of the default Java TrustStores, then do one of the following:
   - Create a TrustStore and configure the connector to use it:
     a. Create a TrustStore containing your signed, trusted server certificate.
     b. Set the `SSLTrustStore` property to the full path of the TrustStore.
     c. Set the `SSLTrustStorePwd` property to the password for accessing the TrustStore.
     d. If the TrustStore is not a JKS TrustStore, set the `SSLTrustStoreType` property to the correct type.
     e. To specify a Java Security API provider, set the `SSLTrustStoreProvider` property to the name of the provider.
   - Or, create a KeyStore and configure the connector to use it:

      a. Create a KeyStore containing your signed, trusted server certificate.

      b. Set the `SSLKeyStore` property to the full path of the KeyStore.

      c. Set the `SSLKeyStorePwd` property to the password for accessing the KeyStore.

      d. If the KeyStore is not a JKS KeyStore, set the `SSLKeyStoreType` property to the correct type.

      e. To specify a Java Security API provider, set the `SSLKeyStoreProvider` property to the name of the provider.

3. Optionally, to allow the SSL certificate used by the server to be self-signed, set the `AllowSelfSignedCerts` property to `1`.

> ⚠️ **Important:**
>
> When the `AllowSelfSignedCerts` property is set to `1`, SSL verification is disabled. The connector does not verify the server certificate against the trust store, and does not verify if the server's host name matches the common name or subject alternative names in the server certificate.

4. Optionally, to allow the common name of a CA-issued certificate to not match the host name of the Impala server, set the `CAIssuedCertNamesMismatch` property to `1`.

For example, the following connection URL connects to a data source using username and password (LDAP) authentication, with SSL enabled:

```
jdbc:impala://localhost:21050;AuthMech=3;SSL=1;
SSLKeyStore=C:\\Users\\bsmith\\Desktop\\keystore.jks;SSLKeySt
orePwd=amazonSSL123;UID=impala;PWD=amazon123
```

> ⓘ **Note:**
>
> For more information about the connection properties used in SSL connections, see

Configuring Server-Side Properties

When connecting to a server that is running Impala 2.0 or later, you can use the connector to apply configuration properties to the server by setting the properties in the connection URL.

> ⚠️ **Important:**
>
> This feature is not supported for earlier versions of Impala, where the SET statement can only be executed from within the Impala shell.

For example, to set the `MEM_LIMIT` query option to 1 GB and the `REQUEST_POOL` query option to `myPool`, you would use a connection URL such as the following:

```
jdbc:impala://localhost:18000/default2;AuthMech=3;
UID=amazon;PWD=amazon;MEM_LIMIT=1000000000;REQUEST_
POOL=myPool
```

Configuring Cloudera Altus Dynamic Service Discovery

You can configure the connector to discover Impala services through Altus. When service discovery is enabled, the connector connects to Impala servers that are part of an Altus cluster.

To enable service discovery, specify the name of the Altus cluster where the Impala services are hosted.

To configure dynamic service discovery through Altus:

1. Create a connection URL that uses the following format, where *[ClusterName]* is the name of the Altus cluster on which Impala services are hosted:

   ```
   jdbc:impala://[ClusterName]
   ```

2. Optionally, to configure authentication, do the following:

   a. Set the `AltusCredFile` property to the full path of the directory where your credentials file is stored.

   b. Set the `AltusProfileName` property to the name of the profile that you want to use for authentication.

3. By default, Altus service discovery uses a public IP address unless the list-cluster-instances API returns "none" for the public IP address. To always use a private IP address, set the `AltusUsePrivateIP` property to `true`.

> **ⓘ Note:**
>
> To make sure that the connection URL is compatible with all JDBC applications, escape the backslashes (\) in your directory paths by typing another backslash.

For example:

```
jdbc:impala://AltusClusterForAmazon
;AltusCredFile=C:\\Documents\\AltusCredentialsFiles;AltusProf
ileName=jsmith;AltusUsePrivateIP=true;
```

For more information about the syntax of the connection URL, see Building the Connection URL on page 13.

Configuring Logging

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

> ⚠ **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 Amazon JDBC Connector for Apache Impala, 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 Amazon JDBC Connector for Apache Impala, in order from least verbose to most verbose.

LogLevel Value	Description
0	Disable all logging.
1	Log severe error events that lead the connector to abort.
2	Log error events that might allow the connector 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 connector.
5	Log detailed information that is useful for debugging the connector.
6	Log all connector activity.

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:impala://localhost:11000;LogLevel=3;LogPath=C:\\temp
   ```

3. To make sure that the new settings take effect, restart your JDBC application and reconnect to the server.

The Amazon JDBC Connector for Apache Impala produces the following log files in the location specified in the `LogPath` property:

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

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

To disable logging:

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

Features

More information is provided on the following features of the Amazon JDBC Connector for Apache Impala:

SQL Translation

The Amazon JDBC Connector for Apache Impala is able to parse queries locally prior to sending them to the Impala server. This feature allows the connector to calculate query metadata without executing the query, support query parameters, and support extra SQL features such as JDBC escape sequences and additional scalar functions that are not available in the Impala-shell tool.

> ℹ **Note:**
>
> The connector does not support translation for queries that reference a field contained in a nested column (an ARRAY, MAP, or STRUCT column). To retrieve data from a nested column, make sure that the query is written in valid Impala SQL syntax.

Data Types

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

The following table lists the supported data type mappings.

Impala Type	SQL Type	Java Type
ARRAY	VARCHAR	String
BIGINT	BIGINT	java.math.BigInteger

Impala Type	SQL Type	Java Type
BINARY	VARBINARY	byte[]
BOOLEAN	BOOLEAN	Boolean
CHAR (Available only in CDH 5.2 or later)	CHAR	String
DATE	DATE	java.sql.Date
DECIMAL (Available only in CDH 5.1 or later)	DECIMAL	java.math.BigDecimal
DOUBLE (REAL is an alias for DOUBLE)	DOUBLE	Double
FLOAT	REAL	Float
INT	INTEGER	Long
MAP	VARCHAR	String
SMALLINT	SMALLINT	Integer
STRUCT	VARCHAR	String
TIMESTAMP	TIMESTAMP	java.sql.Timestamp
TINYINT	TINYINT	Short
VARCHAR (Available only in CDH 5.2 or later)	VARCHAR	String

Catalog and Schema Support

The Amazon JDBC Connector for Apache Impala supports both catalogs and schemas to make it easy for the connector to work with various JDBC applications. Since Impala only organizes tables into schemas/databases, the connector provides a synthetic catalog named IMPALA under which all of the schemas/databases are organized. The connector also maps the JDBC schema to the Impala schema/database.

> **ⓘ Note:**
>
> Setting the `CatalogSchemaSwitch` connection property to `1` will cause Impala catalogs to be treated as schemas in the connector as a restriction for filtering.

Write-back

The Amazon JDBC Connector for Apache Impala supports translation for the following syntax:

- INSERT
- CREATE
- DROP

The connector also supports translation for UPDATE and DELETE syntax, but only when querying Kudu tables while connected to an Impala server that is running Impala 2.7 or later.

If the statement contains non-standard SQL-92 syntax, then the connector is unable to translate the statement to SQL and instead falls back to using Impala SQL.

Dynamic Service Discovery using Cloudera Altus

The Amazon JDBC Connector for Apache Impala can be configured to discover Impala services via Cloudera Altus, and connect to servers that are part of an Altus cluster.

For information about configuring this feature, see Configuring Cloudera Altus Dynamic Service Discovery on page 30.

Security and Authentication

To protect data from unauthorized access, some Impala data stores require connections to be authenticated with user credentials or the SSL protocol. The

Amazon JDBC Connector for Apache Impala provides full support for these authentication protocols.

> **ⓘ Note:**
>
> In this documentation, "SSL" indicates both TLS (Transport Layer Security) and SSL (Secure Sockets Layer). The connector supports industry-standard versions of TLS/SSL.

The connector provides mechanisms that allow you to authenticate your connection using the Kerberos protocol, your Impala user name only, or your Impala user name and password. You must use the authentication mechanism that matches the security requirements of the Impala server. For detailed connector configuration instructions, see Configuring Authentication on page 15.

Additionally, the connector supports SSL connections with one-way authentication. If the server has an SSL-enabled socket, then you can configure the connector to connect to it.

It is recommended that you enable SSL whenever you connect to a server that is configured to support it. SSL encryption protects data and credentials when they are transferred over the network, and provides stronger security than authentication alone. For detailed configuration instructions, see Configuring SSL on page 27.

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

> **ⓘ Note:**
>
> The SSL version used for the connection is the highest version that is supported by both the connector and the server, which is determined at connection time.

Multithreading Support

The Amazon JDBC Connector for Apache Impala supports multithreaded processing.

Within a given process, the connector enables multiple threads to access different connections concurrently. For each connection, the connector enables multiple threads to access different statements from the connection concurrently.

Connector Configuration Options

Connector Configuration Options lists and describes the properties that you can use to configure the behavior of the Amazon JDBC Connector for Apache Impala.

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

> **ⓘ Note:**
>
> Property names and values are case-sensitive.

AllowSelfSignedCerts

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the connector allows the server to use self-signed SSL certificates.

- `1`: The connector allows self-signed certificates.

  > **⚠ Important:**
  >
  > When this property is set to `1`, SSL verification is disabled. The connector does not verify the server certificate against the trust store, and does not verify if the server's host name matches the common name in the server certificate.

- `0`: The connector does not allow self-signed certificates.

> **ⓘ Note:**
>
> This property is applicable only when SSL connections are enabled.

AltusCredFile

Default Value	Data Type	Required
`$HOME/.altus/credentials`	String	No

Description

The full path of the directory that contains your credentials file for accessing an Altus cluster.

> **ⓘ Note:**
>
> This property is applicable only when dynamic service discovery through Altus is enabled.

AltusProfileName

Default Value	Data Type	Required
`default`	String	No

Description

The name of the profile in the credentials file that you want to use to access an Altus cluster. See also AltusCredFile on page 38.

> **ⓘ Note:**
>
> This property is applicable only when dynamic service discovery through Altus is enabled.

AltusUsePrivateIP

Default Value	Data Type	Required
`false`	Boolean	No

Description

This property indicates whether Altus Service discovery always uses a private IP address to establish the connection.

- `true`: The connector always uses a private IP address when Altus service discovery is enabled.
- `false`: The connector uses a public IP address when Altus service discovery is enabled, unless the list-cluster-instances API returns "none" for the public IP address.

AsyncExecPollInterval

Default Value	Data Type	Required
10	Integer	No

Description

The time in milliseconds between each poll for the asynchronous query execution status.

"Asynchronous" refers to the fact that the RPC call used to execute a query against Impala is asynchronous. It does not mean that JDBC asynchronous operations are supported.

AuthMech

Default Value	Data Type	Required
Depends on the `transportMode` setting. For more information, see TransportMode on page 51.	Integer	No

Description

The authentication mechanism to use. Set the property to one of the following values:

- `0` for No Authentication.
- `1` for Kerberos.
- `2` for User Name.

- `3` for User Name And Password.
- `12` for Single Sign-On

CAIssuedCertsMismatch

Default Value	Data Type	Required
`0`	Integer	No

Description

This property specifies whether the connector requires the name of the CA-issued SSL certificate to match the host name of the Impala server.

- `0`: The connector requires the names to match.
- `1`: The connector allows the names to mismatch.

> **ⓘ Note:**
>
> This property is applicable only when SSL connections are enabled.

CatalogSchemaSwitch

Default Value	Data Type	Required
`0`	Integer	No

Description

This property specifies whether the connector treats Impala catalogs as schemas or as catalogs.

- `1`: The connector treats Impala catalogs as schemas as a restriction for filtering.
- `0`: Impala catalogs are treated as catalogs, and Impala schemas are treated as schemas.

DefaultStringColumnLength

Default Value	Data Type	Required
`255`	Integer	No

Description

The maximum number of characters that can be contained in STRING columns. The range of `DefaultStringColumnLength` is `0` to `32767`.

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

DelegationUID

Default Value	Data Type	Required
None	String	No

Description

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

httpPath

Default Value	Data Type	Required
None	String	Yes, if `transportMode=http`.

Description

The partial URL corresponding to the Impala server.

The connector forms the HTTP address to connect to by appending the `httpPath` value to the host and port specified in the connection URL. For example, to connect to the HTTP address `http://localhost:10002/cliservice`, you would use the following connection URL:

```
jdbc:impala://localhost:10002;AuthMech=3;transportMode=http;
httpPath=cliservice;UID=jsmith;PWD=amazon123;
```

> 🛈 **Note:**
>
> By default, Impala servers use `cliservice` as the partial URL.

IgnoreTransactions

Default Value	Data Type	Required
0	Boolean	No

Description

This property specifies whether the connector ignores transaction-related operations or returns an error.

- `1`: The connector ignores any transaction related operations and returns success.
- `0`: The connector returns an "operation not supported" error if it attempts to run a query that contains transaction related operations.

KrbAuthType

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies how the connector obtains the Subject for Kerberos authentication.

- `0`: The connector automatically detects which method to use for obtaining the Subject:
  1. First, the connector tries to obtain the Subject from the current thread's inherited AccessControlContext. If the AccessControlContext contains multiple Subjects, the connector uses the most recent Subject.
  2. If the first method does not work, then the connector checks the `java.security.auth.login.config` system property for a JAAS configuration. If a JAAS configuration is specified, the connector uses that information to create a LoginContext and then uses the Subject associated with it.
  3. If the second method does not work, then the connector checks the KRB5_CONFIG and KRB5CCNAME system environment variables for a Kerberos ticket cache. The connector uses the information from the cache to create a LoginContext and then uses the Subject associated with it.

- `1:` The connector checks the `java.security.auth.login.config` system property for a JAAS configuration. If a JAAS configuration is specified, the connector uses that information to create a LoginContext and then uses the Subject associated with it.
- `2:` The connector checks the KRB5_CONFIG and KRB5CCNAME system environment variables for a Kerberos ticket cache. The connector uses the information from the cache to create a LoginContext and then uses the Subject associated with it.

KrbHostFQDN

Default Value	Data Type	Required
None	String	Yes, if `AuthMech=1`.

Description

The fully qualified domain name of the Impala host.

KrbRealm

Default Value	Data Type	Required
Depends on your Kerberos configuration	String	No

Description

The realm of the Impala host.

If your Kerberos configuration already defines the realm of the Impala host as the default realm, then you do not need to configure this property.

KrbServiceName

Default Value	Data Type	Required
None	String	Yes, if `AuthMech=1`.

Description

The Kerberos service principal name of the Impala server.

LogLevel

Default Value	Data Type	Required
0	Integer	No

Description

Use this property to enable or disable logging in the connector 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 Amazon JDBC Connector for Apache Impala, 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 connector to abort.
- `2`: Enable logging on the ERROR level, which logs error events that might still allow the connector 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 connector.
- `5`: Enable logging on the DEBUG level, which logs detailed information that is useful for debugging the connector.
- `6`: Enable logging on the TRACE level, which logs all connector activity.

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

- An `ImpalaJDBC_driver.log` file that logs connector activity that is not specific to a connection.

- An `Impala_connection_[Number].log` file for each connection made to the database, where *[Number]* is a number that identifies each log file. This file logs connector activity that is specific to the connection.

If the `LogPath` value is invalid, then the connector 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 connector 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.

LowerCaseResultSetColumnName

Default Value	Data Type	Required
1	Integer	No

Description

This property specifies the letter case that the connector uses when returning the column name aliases in the `ResultSetMetadata`.

- `1`: The column name aliases in the `ResultSetMetadata` are returned in lower-case characters, matching the server-side behavior.
- `0`: The column name aliases are returned in the same letter case as specified in the query.

OptimizedInsert

Default Value	Data Type	Required
1	Integer	No

Description

This property specifies whether the connector tries to optimize INSERT statements by bypassing translation.

Each time the connector translates an INSERT statement, it executes the DESCRIBE command to identify the data types of the columns that it is inserting data into. These additional commands consume resources and might reduce connector performance.

- `1`: The connector tries to optimize INSERT statements by bypassing translation and using other methods to identify column types.
- `0`: The connector does not attempt the optimization, and translates INSERT statements normally.

> **ⓘ Note:**
>
> If the optimization fails, the connector falls back to translating INSERT statements normally. This additional overhead might further reduce connector performance.

PreparedMetaLimitZero

Default Value	Data Type	Required
1	Integer	No

Description

This property specifies whether the `PreparedStatement.getMetadata()` call will request metadata from the server with `LIMIT 0`, increasing performance.

- `1`: The `PreparedStatement.getMetadata()` call uses `LIMIT 0`.
- `0`: The `PreparedStatement.getMetadata()` call does not use `LIMIT 0`.

PWD

Default Value	Data Type	Required
None	String	Yes, if `AuthMech=3`.

Description

The password corresponding to the user name that you provided using the property UID on page 52.

RowsFetchedPerBlock

Default Value	Data Type	Required
`10000`	Integer	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	Data Type	Required
`0`	Integer	No

Description

The number of seconds that the TCP socket waits for a response from the server before raising an error on the request.

When this property is set to `0`, the connection does not time out.

SSL

Default Value	Data Type	Required
`0`	Integer	No

Description

This property specifies whether the connector communicates with the Impala server through an SSL-enabled socket.

- `1`: The connector connects to SSL-enabled sockets.
- `2`: The connector connects to SSL-enabled sockets using two-way authentication.
- `0`: The connector does not connect to SSL-enabled sockets.

> **ⓘ Note:**
>
> SSL is configured independently of authentication. When authentication and SSL are both enabled, the connector performs the specified authentication method over an SSL connection.

SSLKeyStore

Default Value	Data Type	Required
None	String	No

Description

The full path of the Java KeyStore containing the server certificate for one-way SSL authentication.

See also the property SSLKeyStorePwd on page 49.

> **ⓘ Note:**
>
> The Amazon JDBC Connector for Apache Impala accepts TrustStores and KeyStores for one-way SSL authentication. See also the property SSLTrustStore on page 49.

SSLKeyStorePwd

Default Value	Data Type	Required
None	Integer	Yes, if you are using a KeyStore for connecting over SSL.

Description

The password for accessing the Java KeyStore that you specified using the property SSLKeyStore on page 48.

SSLTrustStore

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

Description

The full path of the Java TrustStore containing the server certificate for one-way SSL authentication.

If the trust store requires a password, provide it using the property `SSLTrustStorePwd`. See SSLTrustStorePwd on page 50.

> 🛈 **Note:**
>
> The Amazon JDBC Connector for Apache Impala accepts TrustStores and KeyStores for one-way SSL authentication. See also the property SSLKeyStore on page 48.

SSLTrustStorePwd

Default Value	Data Type	Required
None	String	Yes, if using a TrustStore.

Description

The password for accessing the Java TrustStore that you specified using the property SSLTrustStore on page 49.

SSLTrustStoreType

Default Value	Data Type	Required
JKS	String	No

Description

The type of Java TrustStore that is being used for one-way SSL authentication.

SSOWebServerTimeout

Default Value	Data Type	Required
120	Integer	No

Description

This property specifies the number of seconds that the connector waits before timing out while waiting for a browser response when authenticating using Single Sign-On (SSO).

If this property is set to 0, the connector will wait for an indefinite amount of time

StripCatalogName

Default Value	Data Type	Required
1	Integer	No

Description

This property specifies whether the connector removes catalog names from query statements if translation fails or if the `UseNativeQuery` property is set to `1`.

- `1`: If query translation fails or if the `UseNativeQuery` property is set to `1`, then the connector removes catalog names from the query statement.
- `0`: The connector does not remove catalog names from query statements.

SupportTimeOnlyTimestamp

Default Value	Data Type	Required
1	Integer	No

Description

This property specifies whether the connector supports TIMESTAMP data that only contains a time value.

- `1`: The connector supports TIMESTAMP data that only contains a time value.
- `0`: The connector returns an error when working with TIMESTAMP data that only contains a time value.

TransportMode

Default Value	Data Type	Required
sasl	String	No

Description

The transport protocol to use in the Thrift layer.

- `binary`: The connector uses the Binary transport protocol.

  If you use this setting and do not specify the `AuthMech` property, then the connector uses `AuthMech=0` by default. This setting is valid only when the `AuthMech` property is set to `0` or `3`.
- `sasl`: The connector uses the SASL transport protocol.

If you use this setting but do not specify the `AuthMech` property, then the connector uses `AuthMech=2` by default. This setting is valid only when the `AuthMech` property is set to `1`, `2`, or `3`.

- `http`: The connector uses the HTTP transport protocol.

  If you use this setting but do not specify the `AuthMech` property, then the connector uses `AuthMech=0` by default. This setting is valid only when the `AuthMech` property is set to `0`, `3`, or `12`.

> 🛈 **Note:**
>
> This option replaces and supersedes the deprecated `UseSasl` option.

UID

Default Value	Data Type	Required
anonymous	String	**Yes**, if `AuthMech=3`.

Description

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

UpperCaseResultSetColName

Default Value	Data Type	Required
false	Boolean	No

Description

This property specifies whether the connector converts the result set column name to upper case if translation fails or if the `UseNativeQuery` property is set to `1`.

- `true`: If query translation fails or if the `UseNativeQuery` property is set to `1`, the connector converts the result set column names to upper-case characters.
- `false`: The connector does not convert the result set column names to upper-case characters.

UseNativeQuery

Default Value	Data Type	Required
0	Integer	No

Description

This property specifies whether the connector transforms the queries emitted by applications.

- `1`: The connector does not transform the queries emitted by applications, so the native query is used.
- `0`: The connector transforms the queries emitted by applications and converts them into an equivalent form in Impala SQL.

> **ⓘ Note:**
>
> If the application is Impala-aware and already emits Impala SQL, then enable this option to avoid the extra overhead of query transformation.

UseSasl (deprecated)

Default Value	Data Type	Required
1	Integer	No

Description

This option is deprecated. Use `TransportMode` instead (see).

This property indicates if SASL is used in conjunction with the User Name and Password Authentication Mechanism (`AuthMech=3`).

- `0`: No SASL authentication is used. User credentials are still passed to the server for services such as Sentry.
- `1`: SASL authentication is used.

Third-Party Trademarks

Simba, the Simba logo, SimbaEngine, SimbaEngine C/S, SimbaExpress and SimbaLib are registered trademarks of Simba Technologies Inc.

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

Apache Impala, Apache, and Impala are either registered trademarks or trademarks of The Apache Software Foundation in the United States and other countries.

All other trademarks are trademarks of their respective owners.