insightsoftware

Simba Amazon Athena ODBC Data Connector

Installation and Configuration Guide

Version 1.3.3.1000

October 2025

Contents

Copyright

Contact Us

www.insightsoftware.com

About This Guide

Purpose

The *Simba Amazon Athena ODBC Data Connector Installation and Configuration Guide* explains how to install and configure the Simba Amazon Athena ODBC Data Connector. The guide also provides details related to features of the connector.

Audience

The guide is intended for end users of the Simba Amazon Athena ODBC Connector, as well as administrators and developers integrating the connector.

Knowledge Prerequisites

To use the Simba Amazon Athena ODBC Connector, the following knowledge is helpful:

- Familiarity with the platform on which you are using the Simba Amazon Athena ODBC Connector

- Ability to use the data source to which the Simba Amazon Athena ODBC Connector is connecting

- An understanding of the role of ODBC technologies and driver managers in connecting to a data source

- Experience creating and configuring ODBC connections

- Exposure to SQL

Document Conventions

Italics is 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.

About the Simba Amazon Athena ODBC Connector

About Amazon Athena

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

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

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

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

About the Connector

The Simba Amazon Athena ODBC Connector enables organizations to connect their BI tools to the Amazon Athena query service, enabling Business Intelligence, analytics, and reporting on the data that Athena returns from Amazon S3 databases. The connector retrieves catalog metadata from the AthenaAPI.

The connector complies with the ODBC 3.80 data standard, including important functionality such as Unicode and 32- and 64-bit support for high-performance computing environments on all platforms.

ODBC 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 ODBC connector, which connects an application to the database. For more information about ODBC, see *Data Access Standards* on the Simba Technologies website: https://www.simba.com/resources/data-access-standards-glossary. For complete information about the ODBC specification, see the *ODBC API Reference* from the Microsoft documentation: https://docs.microsoft.com/en-us/sql/odbc/reference/syntax/odbc-api-reference.

The Simba Amazon Athena ODBC Connector is tested on the following data sources:

- AWS Glue Data Catalog (default)

- Apache Hive metastore

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

> **Note:**
> For information about how to use the connector in various BI tools, see the *Simba ODBC Connectors Quick Start Guide for Windows*: http://cdn.simba.com/docs/ODBC_ QuickstartGuide/content/quick_start/intro.htm.

Windows Connector

This section provides an overview of the Connector in the Windows platform, outlining the required system specifications and the steps for installing and configuring the connector in Windows environments.

Windows System Requirements

Install the connector on client machines where the application is installed. Before installing the connector, make sure that you have the following:

- Administrator rights on your machine.

- A machine that meets the following system requirements:
    - One of the following operating systems:
        - Windows 10, 8.1, or 7 SP1
        - Windows Server 2025, 2016, 2012, or 2008 R2 SP1
    - 150 MB of available disk space

Before the connector can be used, the Visual C++ Redistributable for Visual Studio 2022 with the same bitness as the connector must also be installed. If you obtained the connector from the Simba website, then your installation of the connector automatically includes this dependency. Otherwise, you must install the redistributable manually. You can download the installation packages for the redistributable at https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170.

Installing the Connector in Windows

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Connector Installation Guide*.

On 64-bit Windows operating systems, you can execute both 32-bit and 64-bit applications. However, 64-bit applications must use 64-bit connectors, and 32-bit applications must use 32-bit connectors. Make sure that you use a connector whose bitness matches the bitness of the client application:

- `Simba Athena 1.3 32-bit.msi` for 32-bit applications

- `Simba Athena 1.3 64-bit.msi` for 64-bit applications

You can install both versions of the connector on the same machine.

To install the Simba Amazon Athena ODBC Connector in Windows:

1. **1.3 32-bit.msi** or **Simba Athena 1.3 64-bit.msi**.

2. Depending on the bitness of your client application, double-click to run **Simba Athena 1.3 32-bit.msi** or **Simba Athena 1.3 64-bit.msi**.

3. Click **Next**.

4. Select the check box to accept the terms of the License Agreement if you agree, and then click **Next**.

5. To change the installation location, click **Change**, then browse to the desired folder, and then click **OK**. To accept the installation location, click **Next**.

6. Click **Install**.

7. When the installation completes, click **Finish**.

8. If you received a license file through email, then copy the license file into the `\lib` subfolder of the installation folder you selected above. You must have Administrator privileges when changing the contents of this folder.

Creating a Data Source Name in Windows

Typically, after installing the Simba Amazon Athena ODBC Connector, you need to create a Data Source Name (DSN).

Alternatively, for information about DSN-less connections, see Using a Connection String.

To create a Data Source Name in Windows:

1. From the Start menu, go to **ODBC Data Sources**.

   > ℹ️ **Note:** Make sure to select the ODBC Data Source Administrator that has the same bitness as the client application that you are using to connect to Athena.

2. In the ODBC Data Source Administrator, click the **Drivers** tab, and then scroll down as needed to confirm that the Simba Amazon Athena ODBC Connectorappears in the alphabetical list of ODBC Drivers that are installed on your system.

3. Choose one:

   - To create a DSN that only the user currently logged into Windows can use, click the **User DSN** tab.

   - Or, to create a DSN that all users who log into Windows can use, click the **System DSN** tab.

     > ℹ️ **Note:** It is recommended that you create a System DSN instead of a User DSN. Some applications load the data using a different user account, and might not be able to detect User DSNs that are created under another user account.

4. Click **Add**.

5. In the Create New Data Source dialog box, select **Simba Amazon Athena ODBC Connector**and then click **Finish**. The Simba Amazon Athena ODBC ConnectorDSN Setup dialog box opens.

6. In the **Data Source Name** field, type a name for your DSN.

7. Optionally, in the **Description** field, type relevant details about the DSN.

8. In the **AWS Region** field, type the AWS region of the Athena instance that you want to connect to.

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

9. Optionally, in the **Catalog** field, type the name of the Athena catalog you want the connector to query. If lthis filed is left empty, queries will be applied to AwsDataCatalog.

10. In the **Schema** field, type the name of the database schema to use when a schema is not explicitly specified in a query. You can still issue queries on other schemas by explicitly specifying the schema in the query.

11. Optionally, in the **Workgroup** field, type the name of the workgroup to use when signing in to Athena.

12. Optionally, from the **Metadata Retrieval Method** drop-down list, select the option you want to use to retrieve metadata from Athena:

Option Name	Description
Auto	At connection time connector automatically determines whether to use AWS Glue or Query to retrieve metadata for the specified Athena region. If AWS Glue is supported in the region and Athena has been upgraded to use AWS Glue, the connector uses AWS Glue to retrieve the metadata. If AWS Glue is not supported in the region or Athena hasn't been upgraded to use AWS Glue, the connector queries Athena to retrieve the metadata.
Glue	The connector uses AWS Glue to retrieve the metadata regardless of whether AWS Glue is supported or used in the region.
ProxyAPI	The connector uses Athena's proxy API. This is used to query external catalogs.
Query	The connector uses Query to retrieve the metadata regardless of whether AWS Glue is supported or used in that region.

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

13. In the **S3 Output Location** field, type the path of the Amazon S3 location where you want to store query results, prefixed by `s3://`.

    For example, to store results in a folder named "test-folder-1" inside an S3 bucket named "query-results-bucket", you would type **s3://query-results-bucket/test-folder-1** in this field.

14. To configure encryption for your query results, do the following:
    a. From the **Encryption Options** drop-down list, select the encryption protocol that you want to use:

Option Name	Description
NOT_SET	The connector does not encrypt the data.
SSE_S3	The connector uses server-side encryption with an Amazon S3-managed key.
SSE_KMS	The connector uses server-side encryption with an AWS KMS-managed key.
CSE_KMS	The connector uses client-side encryption with an AWS KMS-managed key.

For detailed information about these encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*:
http://docs.aws.amazon.com/athena/latest/ug/encryption.html.

b. If you selected SSE_KMS or CSE_KMS in the previous step, then in the **KMS Key** field, type the KMS customer key to use for encrypting data.

15. To configure authentication, click **Authentication Options**. For more information, see Configuring Authentication in Windows.

16. To configure advanced options, click **Advanced Options**. For more information, see Configuring Advanced Options in Windows.

17. To configure proxy connections, click **Proxy Options**. For more information, see Configuring Proxy Connections in Windows.

18. To configure logging behavior for the connector, click **Logging Options**. For more information, see Configuring Logging Options in Windows.

19. To test the connection, click **Test**. Review the results as needed, and then click **OK**.

> **Note:** If the connection fails, then confirm that the settings in the Simba Athena ODBC Driver DSN Setup dialog box are correct. Contact your AWS account administrator as needed.

20. To save your settings and close the Simba Athena ODBC Driver DSN Setup dialog box, click **OK**.

21. To close the ODBC Data Source Administrator, click **OK**.

Configuring Authentication in Windows

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

- Using the Default Credentials Provider Chain in Windows

- Using IAM Credentials in Windows

- Using an IAM Profile in Windows

- Using an Instance Profile in Windows

- Using the Active Directory Federation Services (AD FS) Credentials Provider in Windows

- Using the Azure AD Credentials Provider in Windows

- Using the Browser Azure AD Credentials Provider in Windows

- Using the Browser Plugin for a SAML Service in Windows

- Using a JSON Web Token (JWT)

- Using the Okta Service

- Using Okta MFA with Okta Verify

- Using Okta MFA with Google Authenticator

- Using Okta MFA with SMS Authentication

- Using PingFederate Service in Windows

- Using an External Credentials Service

Using the Default Credentials Provider Chain in Windows

You can configure the connector to authenticate the connection using credentials that are stored in one of the locations in the default credentials provider chain. The connector looks for a valid access key and secret key pair by checking the following locations, in the following order:

1. The AWS credentials file stored in the `%USERPROFILE%.awscredentials` directory.

2. The AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY system environment variables.

3. The instance profile from the Amazon EC2 Instance Metadata Service.

For detailed information about configuring default credentials, see "Providing AWS Credentials" in the *AWS SDK for C++ Developer Guide*: http://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/credentials.html.

To configure authentication using the default credentials provider chain in Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Authentication Options**.

2. From the **Authentication Type** drop-down list, select **Default Credentials**.

3. To save your settings and close the Authentication Options dialog box, click **OK**.

Using IAM Credentials in Windows

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

If you are using temporary credentials, which are only valid for a limited amount of time, then you must also provide a session token. For more information, see "Temporary Security Credentials" in the *AWS Identity and Access Management User Guide*: http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html.

To configure authentication using IAM credentials in Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Authentication Options**.

2. From the **Authentication Type** drop-down list, select **IAM Credentials**.

3. In the **User** field, type the access key provided by your AWS account.

4. In the **Password** field, type the secret key provided by your AWS account.

5. To encrypt your credentials, click **Password Options** and then select one of the following:

   - If the credentials are used only by the current Windows user, select **Current User Only**.

   - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

6. If you are using temporary credentials, in the **Session Token** field, type the session token generated by the AWS Security Token Service.

7. Optionally, to retrieve SAML credentials using Lake Formation service, select the **LakeFormation Enabled** checkbox.

8. To save your settings and close the Authentication Options dialog box, click **OK**.

Using an IAM Profile in Windows

You can configure the connector to authenticate the connection using credentials that are associated with an IAM profile in a credentials file.

By default, the connector uses the credentials associated with a profile named `default` in the credentials file found in the `%USERPROFILE%.awscredentials` directory. To use a different profile, specify the profile name in your connection settings. To use a different credentials file, set the AWS_SHARED_CREDENTIALS_FILE system environment variable to the full path of your credentials file.

For information about the format of a credentials file, see the "AWS Credentials File Format" section from the "Working with AWS Credentials" page in the *AWS SDK for Java Developer Guide*: http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html.

To configure authentication using an IAM profile in Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Authentication Options**.

2. From the **Authentication Type** drop-down list, select **IAM Profile**.

3. In the **AWS Profile** field, type the name of the profile to use.

4. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role you want to assume when authenticating through an external credential service.

5. To save your settings and close the Authentication Options dialog box, click **OK**.

Using an Instance Profile in Windows

You can configure the connector to authenticate the connection using credentials that have been loaded from the Amazon EC2 Instance Metadata Service into an instance profile.

Instance profiles contain authorization information such as roles, permissions, and credentials, and are automatically created by Amazon EC2 for each IAM role that is defined for an EC2 instance. For more information, see "IAM Roles for Amazon EC2" in the *Amazon Elastic Compute Cloud User Guide for Windows Instances*: http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/iam-roles-for-amazon-ec2.html.

To configure authentication using an instance profile in Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Authentication Options**.

2. From the **Authentication Type** drop-down list, select **Instance Profile**.

3. To save your settings and close the Authentication Options dialog box, click **OK**.

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

You can configure the connector to authenticate the connection using credentials obtained from the AD FS credentials provider. To do this, you must specify information about the AD FS service, such as the host and port of the server where the service is hosted.

To configure authentication using AD FS in Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Authentication Options**.

2. From the **Authentication Type** drop-down list, select **ADFS**.

3. Optionally, to specify your credentials for accessing the AD FS server, do the following. If you do not specify any credentials, the connector attempts to authenticate to the AD FS server by using your Windows account credentials over the NTLM protocol.

    a. In the **User** field, type the user name that you use to access the AD FS server. You can include the domain name using the format *[DomainName]\[UserName]*.

    b. In the **Password** field, type the password corresponding to the user name that you provided in the previous step.

    c. To encrypt your credentials, click **Password Options** and then select one of the following:

        - If the credentials are used only by the current Windows user, select **Current User Only**.

        - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

4. To specify AD FS service information, do the following:

    a. In the **IdP Host** field, type the host name of the AD FS service.

        > **Note:** The host name cannot include any slashes (/).

    b. Optionally, in the **IdP Port** field, type the number of the port that the AD FS service host uses to listen for requests.

> **Note:** The exact port number that you need to specify may differ depending on the AD FS server configuration. If you are not sure which port to specify, contact your system administrator.

5. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through AD FS.

6. Optionally, in the **Session Duration** field, type the duration, in seconds, of the role session.

7. If the AD FS service must be accessed through an HTTP proxy, select the **Use HTTP Proxy For IdP Host** check box. For information about configuring the proxy connection, see Configuring Proxy Connections in Windows.

8. Optionally, if you do not want the connector to verify the AD FS server certificate, select the **SSL Insecure** check box.

9. To save your settings and close the Authentication Options dialog box, click **OK**.

Using the Azure AD Credentials Provider in Windows

You can configure the connector to authenticate the connection using credentials obtained from the Azure AD credentials provider. To do this, you must specify information about the Azure AD service, such as the Client ID and Secret and the Tenant ID.

To configure authentication using Azure AD in Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Authentication Options**.

2. From the **Authentication Type** drop-down list, select **AzureAD**.

3. In the **User** field, type the user name that you use to access the Azure AD server.

4. In the **Password** field, type the password corresponding to the user name that you provided in the previous step.

5. To encrypt your credentials, click **Password Options** and then select one of the following:

   - If the credentials are used only by the current Windows user, select **Current User Only**.

   - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

6. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Azure AD.

7. Optionally, in the **Session Duration** field, type the duration, in seconds, of the role session.

8. In the **Tenant ID** field, type the Azure AD-provided unique ID associated with your Athena application.

9. In the **Client ID** field, type the Client ID to use when authenticating the connection using the Azure AD service.

10. In the **Client Secret** field, type the Client Secret to use when authenticating the connection using the Azure AD service.

11. To save your settings and close the Authentication Options dialog box, click **OK**.

Using the Browser Azure AD Credentials Provider in Windows

You can configure the connector to authenticate the connection using credentials obtained from the Azure AD credentials provider. To do this, you must specify information about the Azure AD service, such as the Client ID and Secret and the Tenant ID.

To configure authentication using Azure AD in Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Authentication Options**.

2. From the **Authentication Type** drop-down list, select **BrowserAzureAD**.

3. In the **User** field, type the user name that you use to access the Azure AD server.

4. In the **Password** field, type the password corresponding to the user name that you provided in the previous step.

5. To encrypt your credentials, click **Password Options** and then select one of the following:

   - If the credentials are used only by the current Windows user, select **Current User Only**.

   - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

6. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Azure AD.

7. Optionally, in the **Session Duration** field, type the duration, in seconds, of the role session.

8. In the **Tenant ID** field, type the Azure AD-provided unique ID associated with your Athena application.

9. In the **Client ID** field, type the Client ID to use when authenticating the connection using the Azure AD service.

10. Optionally, in the **Client Secret** field, type the Client Secret to use when authenticating the connection using the Azure AD service.

11. Optionally, in the **Timeout** field, type the maximum amount of time, in seconds, that the connector is to wait for the redirect URI to fetch the authorization code during Browser Azure AD authentication.

12. To save your settings and close the Authentication Options dialog box, click **OK**.

   > **Note:** The redirect URI must be in the following format:
   > `http://localhost:portnumber/athena.`

Using the Browser Plugin for a SAML Service in Windows

You can configure the connector to use a browser plugin to authenticate your connection through a SAML service such as Okta, Ping, or AD FS.

To configure IAM authentication using a browser plugin in Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Authentication Options**.

2. From the **Authentication Type** drop-down list, select **BrowserSAML**.

3. In the **User** field, type the user name that you use to access the Azure AD server.

4. In the **Password** field, type the password corresponding to the user name that you provided in the previous step.

5. To encrypt your credentials, click **Password Options** and then select one of the following:

   - If the credentials are used only by the current Windows user, select **Current User Only**.

   - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

6. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Azure AD.

7. Optionally, in the **Session Duration** field, type the duration, in seconds, of the role session.

8. In the **Login URL** field, type the URL for the resource on the identity provider's website.

9. Optionally, in the **Listen Port** field, type the number of the port that the connector uses to receive the SAML response from the identity provider.

10. Optionally, in the **Timeout (sec)** field, type the amount of time, in seconds, that the connector waits for the SAML response from the identity provider.

11. To save your settings and close the Authentication Options dialog box, click **OK**.

Using a JSON Web Token (JWT)

You can configure the connector to authenticate your connection by using a token obtained from the web identity provider.

To configure IAM authentication using a JWT in Windows:

1. To access the authentication options, open the **ODBC Data Source Administrator** where you created the DSN, select the DSN, and then click Configure.

2. In the Authentication area, click the **Authentication Type** drop-down list, select **JWT**.

3. In the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through JWT.

4. In the **Web Token** field, type the JSON web token generated after OAuth authentication from Azure Active directory.

5. Optionally, in the **Role Session Name** field, type the name of the assumed role session.

6. To save your settings and close the dialog box, click **OK**.

Using the Okta Service

You can configure the connector to authenticate the connection using credentials obtained from the Okta credentials provider. To do this, you must specify information about the Okta service, such as the host name of the Okta service and the Okta application ID.

To configure authentication using Okta in Windows:

1. To access the authentication options, open the **ODBC Data Source Administrator** where you created the DSN, select the DSN, and then click Configure.

2. In the Authentication area, click the **Authentication Type** drop-down list, select **Okta**.

3. In the **User** field, type the user name associated with your Okta account.

4. In the **Password** field, type the password associated with your Okta user name.

5. In the **IdP Host** field, type the host name of the Okta service.

6. In the **Okta App ID** field, type the Okta-supplied ID associated with your Athena application.

7. Optionally, in the **Okta App Name** field, type the name of your Okta application.

8. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.

9. To save your settings and close the dialog box, click **OK**.

Using Okta MFA with Okta Verify

You can configure the connector to authenticate the connection using credentials obtained from Okta MFA with Okta Verify factor.

> ⚠ **Important:**
> Ensure that the Okta Verify factor is enabled on Okta.

To enroll with Okta Verify with TOTP:

1. Scan the QR code displayed on the browser using the Okta Verify app.

2. In the dialog box, enter the password from the app.

3. Activate the device.

To enroll with Okta Verify with push:

1. Scan the QR code displayed on the browser using the Okta Verify app.

2. Approve the push notification sent to your device.

To configure authentication using Okta Verify factor in Windows:

1. To access the authentication options, open the **ODBC Data Source Administrator** where you created the DSN, select the DSN, and then click Configure.

2. In the Authentication area, click the **Authentication Type** drop-down list, select **Okta**.

3. In the **User** field, type the user name associated with your Okta account.

4. In the **Password** field, type the password associated with your Okta user name.

5. In the **IdP Host** field, type the host name of the Okta service.

6. In the **Okta App ID** field, type the Okta-supplied ID associated with your Athena application.

7. Optionally, in the **Okta App Name** field, type the name of your Okta application.

8. Optionally, in the **Okta MFA wait time** field, type the MFA timeout value, in seconds.

9. In the **Okta MFA Type** field, type either `oktaverifywithtotp` or `oktaverifywithpush`.

10. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.

11. To save your settings and close the dialog box, click **OK**.

The following is an example connection string that uses Okta Verify with push:

UID=Scooby;PWD=Scrappy;idp_port=443;idp_host=dev-580551.okta.com;app_id=0oare1348E1NyLw42356/272;okta_mfa_type=oktaverifywithpush;

Using Okta MFA with Google Authenticator

You can configure the connector to authenticate the connection using credentials obtained from Okta MFA with Google Authenticator.

> ⚠ **Important:**
> Ensure that the Google Authenticator factor is enabled on Okta.

To enroll with Okta Google Authenticator:

1. Scan the QR code displayed on the browser using the Okta Verify app.

2. In the dialog box, type the password from the app.

3. Activate the device.

To configure authentication using Okta MFA with Google Authenticator in Windows:

1. To access the authentication options, open the **ODBC Data Source Administrator** where you created the DSN, select the DSN, and then click Configure.

2. In the Authentication area, click the **Authentication Type** drop-down list, select **Okta**.

3. In the **User** field, type the user name associated with your Okta account.

4. In the **Password** field, type the password associated with your Okta user name.

5. In the **IdP Host** field, type the host name of the Okta service.

6. In the **Okta App ID** field, type the Okta-supplied ID associated with your Athena application.

7. Optionally, in the **Okta App Name** field, type the name of your Okta application.

8. Optionally, in the **Okta MFA wait time** field, type the MFA timeout value, in seconds.

9. In the **Okta MFA Type** field, type `googleauthenticator`.

10. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.

11. To save your settings and close the dialog box, click **OK**.

The following is an example connection string that uses Okta MFA with Google Authenticator:

UID=Scooby;PWD=Scrappy;idp_port=443;idp_host=dev-580551.okta.com;app_
id=0oare1348E1NyLw42356/272;okta_mfa_type=GoogleAuthenticator;

Using Okta MFA with SMS Authentication

You can configure the connector to authenticate the connection using credentials obtained from Okta MFA with SMS Authentication.

> **!** **Important:**
> Ensure that the SMS Authentication factor is enabled on Okta.

To enroll with Okta SMS Authentication:

- Enter the password sent to your mobile device.

To configure authentication using Okta MFA with SMS Authentication in Windows:

1. To access the authentication options, open the **ODBC Data Source Administrator** where you created the DSN, select the DSN, and then click Configure.

2. In the Authentication area, click the **Authentication Type** drop-down list, select **Okta**.

3. In the **User** field, type the user name associated with your Okta account.

4. In the **Password** field, type the password associated with your Okta user name.

5. In the **IdP Host** field, type the host name of the Okta service.

6. In the **Okta App ID** field, type the Okta-supplied ID associated with your Athena application.

7. Optionally, in the **Okta App Name** field, type the name of your Okta application.

8. Optionally, in the **Okta MFA wait time** field, type the MFA timeout value, in seconds.

9. In the **Okta MFA Type** field, type `smsauthentication`.

10. In the **Okta MFA Phone No** field, type a US or Canadian number.

11. Optionally, in the **Preferred Role** field, type the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.

12. To save your settings and close the dialog box, click **OK**.

The following is an example connection string that uses Okta MFA with SMS Authentication:

UID=Scooby;PWD=Scrappy;idp_port=443;idp_host=dev-580551.okta.com;app_
id=0oare1348E1NyLw42356/272;okta_mfa_type=SmsAuthentication;okta_phone_number=*[US or
Canadian Number]*;

> **Note:** Okta only supports a US or Canadian number. If a number is not preceded with the country code, +1 is automatically added.

Using PingFederate Service in Windows

You can configure the connector to authenticate your connection through IAM authentication using the credentials stored in the PingFederate service.

To configure IAM authentication using PingFederate service in Windows:

1. To access the IAM authentication options, open the ODBC Data Source Administrator where you created the DSN, select the DSN, and then click **Configure**.

2. From the **Authentication Type** drop-down list, select **Ping**.

3. In the **User** field, type the user name associated with your Ping account.

4. In the **Password** field, type the password associated with your Ping user name.

5. In the **Preferred Role** field, type the name or ID for the IAM role you want the user to assume when logged in to Athena.

6. In the **IdP Host** field, type the address of the service host.

7. In the **IdP Port** field, type the port number the service listens at.

8. To skip verification of the SSL certificate of the IDP server, select the **SSL Insecure** check box.

9. Optionally, in the **Partner SPID** field, type a partner SPID (service provider ID) value.

10. To save your settings and close the dialog box, click **OK**.

Using an External Credentials Service

In addition to built-in support for AD FS, Azure AD, and Okta, the Windows version of the Simba Amazon Athena ODBC Connector also provides support for other credentials services. The connector can authenticate connections using any SAML-based credential provider plugin of your choice.

To configure an external credentials service in Windows:

1. Create an IAM profile that specifies the credential provider plugin and other authentication parameters as needed. The profile must be ASCII-encoded, and must contain the following key-value pair, where *[PluginPath]* is the full path to the plugin application:

   plugin_name = *[PluginPath]*

   For example:

   plugin_name = C:\Users\jsmith\ApplicationInstallDir\CredServiceApplication.exe

For information about how to create a profile, see "Using a Configuration Profile" in the *Amazon Redshift Cluster Management Guide*: https://docs.aws.amazon.com/redshift/latest/mgmt/options-for-providing-iam-credentials.html#using-configuration-profile.

2. Configure the connector to use this profile. The connector detects and uses the authentication settings specified in the profile.

The following code sample shows how to create the `.exe` file for an external credential service:

```cpp
#include "SampleIAMWinHttpClient.h"

#include <aws/core/Aws.h>

#include <aws/core/http/HttpTypes.h>

#include <aws/core/utils/json/JsonSerializer.h>

#include <map>

#include <regex>

#include <string>

#include <Windows.h>

#include <winhttp.h>

/* Json::JsonValue class contains a member function: GetObject. There is a predefined

* MACRO GetObject in wingdi.h that will cause the conflict. We need to undef GetObject

* in order to use the GetObject memeber function from Json::JsonValue

*/
#ifdef GetObject

        #undef GetObject

#endif


using namespace std;

using namespace Aws::Client;

using namespace Aws::Http;

using namespace Aws::Utils;


namespace

{

        static const string IAM_KEY_IDP_HOST = "idp_host"

        static const string IAM_KEY_APP_ID = "app_id";

        static const string IAM_KEY_APP_NAME = "app_name";
```

```cpp
        static const string IAM_KEY_USER = "user";

        static const string IAM_KEY_PASSWORD = "password";


        static const string OKTA_SESSION_TOKEN_STATUS = "status";

        static const string OKTA_SESSION_TOKEN_STATUS_SUCCESS =
        "SUCCESS";

        static const string OKTA_SESSION_TOKEN = "sessionToken";
}
/*/////////////////////////////////////////////
// @brief Replace all instances of findString with
// replaceWith.
//
// @param source The original string and return as a
// modified string.
// @param oldValue The value to be replaced.
// @param newValue The value to use as a replacement.
/////////////////////////////////////////////*/
void FindAndReplace(string& source, string& oldValue, string& newValue)
{
        size_t pos = source.find(oldValue);

        while (pos != string::npos)
        {
                source.replace(pos, oldValue.length(), newValue);

                pos = source.find(oldValue, (pos + newValue.length()));
        }
        return;
}
/*/////////////////////////////////////////////
// @brief Get the list of connection and
// authentication parameters (key-value pairs) from
// the AWS profile. The connector will convert them into
// [key]=[value] arguments and pass to this program.
```

```
// Sample command line:

// IAMExternalPlugin.exe idp_

// host=sampleidphost.okta.com app_id=sampleappid app_

// name=amazon_aws

// user=sampleuser password=samplepassword

//

// @param argc The number of command-line arguments.

// @param argv The argument vector.

//

// @return A list of key-value pairs used to connect

// and authenticate an Okta server.

/////////////////////////////////////////////*/

std::map<std::string, std::string> GetArgsList(int argc, char* argv[])

{

        map<string, string> result;

        const string DELIMITER = "=";

        for (int i = 0; i < argc; ++i)

        {

                string arg(argv[i]);

                size_t pos = arg.find(DELIMITER);

                if (string::npos != pos)

                {

                        string key = arg.substr(0, pos);

                        string value = arg.substr(pos + 1);

                        result[key] = value;

                }

        }

        return result;

}

/*/////////////////////////////////////////

// @brief Create a WinHttp client with AWS client
```

```cpp
// configuration.
// SampleIAMWinHttpClient is implemented using the
// AWS SDK for C++.
// You can use any HTTP client of your choice.
//
// @param config The AWS client configuration.
//
// @return SampleIAMWinHttpClient.
//////////////////////////////////////*/
std::shared_ptr<SampleIAMWinHttpClient> GetHttpClient(const ClientConfiguration& config)
{
        return Aws::MakeShared<SampleIAMWinHttpClient>
        ("SampleIAMWinHttpClient", config);
}
/*//////////////////////////////////////////
// @brief Get the SAML response, in base64-encoded
// ASCII string, returned by the specific
// SAML provider being used for this implementation.
// How you get this string will depend on
// the specific SAML provider you are using. A sample
// implementation of OKTA credentials
// provider is given below.
//
// @param args The list of connection and
// authentication parameters.
//
// @return The SAML response string.
//////////////////////////////////////////*/
std::string GetSamlResponse(std::map<std::string, std::string> args)
{
        string samlResponse;
```

```cpp
// Initialize AWS SDK before creating service
// clients and using them
Aws::SDKOptions options;
Aws::InitAPI(options);


// The URL endpoint for OKTA authentication
string uri = "https://" + args[IAM_KEY_IDP_HOST] +
"/api/v1/authn";


// HTTP Request header
const map<string, string> requestHeaders =
{
        { "Accept", "application/json" },
        { "Content-Type", "application/json; charset=utf-8" },
        { "Cache-Control", "no-cache" }
 };
// OKTA authentication parameters
const map<string, string> requestParamMap =
{
        { "username", args[IAM_KEY_USER] },
        { "password", args[IAM_KEY_PASSWORD] }
};
// Create a HTTP request body from the
// authentication parameters
Json::JsonValue requestBodyJson;
for (const auto& param : requestParamMap)
{
        requestBodyJson.WithString(param.first,
        param.second);
}
```

```cpp
Json::JsonView requestBodyView(requestBodyJson);
const string requestBody = requestBodyView.WriteReadable();


        // Create an IAMHttpClient with HttpClient
        // configuration.
        ClientConfiguration config;
        std::shared_ptr<SampleIAMWinHttpClient> client = GetHttpClient
        (config);


        // Send HTTP POST request to the OKTA server for
        // authentication
        std::shared_ptr<Aws::Http::HttpResponse> response = client-
        >MakeHttpRequest(
                uri,
                HttpMethod::HTTP_POST,
                requestHeaders,
                requestBody);


        // Check the response code from the HTTP POST
        // request
        if (HttpResponseCode::OK != response->GetResponseCode())
        {
        cerr << "Connection or authentication failed to the Okta server.
        Response code: ";
        cerr << static_cast<int>(response->GetResponseCode()) << endl;
        return samlResponse;
        }


        // Parse the response body to a JSON document
        Aws::Utils::Json::JsonValue jsonValue(response-
        >GetResponseBody());
```

```
        if (!jsonValue.WasParseSuccessful())

        {

        cerr << "Failed to parse the response into JSON: ";

        cerr << jsonValue.GetErrorMessage() << endl;

        return samlResponse;

        }
// Check whether the status of the session token
// request succeeded
        string status;

        string sessionToken;
Json::JsonView jsonNodeView(jsonValue);
if (jsonNodeView.ValueExists(OKTA_SESSION_TOKEN_STATUS) &&
jsonNodeView.GetObject(OKTA_SESSION_TOKEN_STATUS).IsString())
        {

        status = jsonNodeView.GetString(OKTA_SESSION_TOKEN_
        STATUS);

        }
// Extract the session token from the response
        if (OKTA_SESSION_TOKEN_STATUS_SUCCESS == status)

        {

        if (jsonNodeView.ValueExists(OKTA_SESSION_TOKEN) &&
        jsonNodeView.GetObject(OKTA_SESSION_TOKEN).IsString())

                {

                sessionToken = jsonNodeView.GetString(OKTA_
                SESSION_TOKEN);

                }

        }


        if (sessionToken.empty())

        {

        cerr << "Failed to retrieve Okta session token." << endl;

        return samlResponse;
```

```
        }


        // Sending HTTP GET request using the session
        // token to get the SAML response from server
        uri = "https://" + args[IAM_KEY_IDP_HOST] + "/home/" + args
        [IAM_KEY_APP_NAME] + "/" + args[IAM_KEY_APP_ID] +
        "?onetimetoken=" + sessionToken;

        std::shared_ptr<Aws::Http::HttpResponse> responseGet = client-
        >MakeHttpRequest(uri, HttpMethod::HTTP_GET);


        if (HttpResponseCode::OK != responseGet->GetResponseCode())

        {

        cerr << "Failed to retrieve SAML assertion from the Okta server.
        Response code: ";

        cerr << static_cast<int>(response->GetResponseCode()) << endl;

        return samlResponse;

        }


        // Extract the SAML response from the HTTP
        // response
        Aws::StringStream reponseBody;

        reponseBody << responseGet->GetResponseBody().rdbuf();

        string resBody = reponseBody.str();


        std::smatch match;

        std::regex expression("SAMLResponse.+?value=\"([^\"]+)\"");

        if (!std::regex_search(resBody, match, expression))

        {

        cerr << "SAML assertion not found." << endl;

        return samlResponse;

        }
```

```
                samlResponse = match.str(1);


                string findString = "&#x2b;";
                string replaceWith = "+";
                FindAndReplace(samlResponse, findString, replaceWith);


                findString = "&#x3d;";
                replaceWith = "=";
                FindAndReplace(samlResponse, findString, replaceWith);


                // Shutdown AWS SDK
                Aws::ShutdownAPI(options);


                // Return the SAML response
                return samlResponse;
}


/*///////////////////////////////////////////////
// @breif The program starts here. The OKTA connection
// and authentication parameters are passed
// from the command line.
//
// @param argc The number of command-line arguments.
// @param argv The argument vector.
///////////////////////////////////////////////*/
void main(int argc, char* argv[])
{
        // Get the list of connection and authentication
        // parameters (key-value pairs) from the AWS
        // profile.
        std::map<std::string, std::string> args(GetArgsList(argc, argv));
```

```
// Get the SAML response in base64-encoded ASCII

// string.

// Example of SAML response:
// https://www.samltool.com/generic_sso_res.php

// TODO: This is the method you need to implement in

// order to get the SAML response string

// returned by the specific SAML provider.

std::string samlResponse = GetSamlResponse(args);

// Send the SAML response string to the StdOutput.

// The Athena ODBC Connector then decodes and

// authenticates the Athena server using the SAML

// response.

std::cout << samlResponse;

}
```

Configuring Advanced Options in Windows

You can configure advanced options to modify the behavior of the connector.

To configure advanced options in Windows:

1. To access advanced options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Advanced Options**.

2. In the **String Column Length** field, type the maximum data length for STRING columns.

3. In the **Binary Column Length** field, type the maximum data length for BINARY columns.

4. In the **Max Complex Type Column Length** field, type the maximum data length for complex data types the connector casts to SQL_VARCHAR.

5. In the **Max Catalog Name Length** field, type the maximum number of characters that catalog names contain.

6. In the **Max Schema Name Length** field, type the maximum number of characters that schema names contain.

7. In the **Max Table Name Length** field, type the maximum number of characters that table names contain.

8. In the **Max Column Name Length** field, type the maximum number of characters that column names contain.

> **Note:** For the options described in steps 4 to 8, you can specify `0` to indicate that there is no maximum length or that the length is unknown.

9. In the **Rows To Fetch Per Block** field, type the maximum number of rows to fetch per stream if using the result set streaming API.

10. In the **Max Execution Polling Interval** field, type the maximum time, in milliseconds, to wait between attempts when polling the server for the query execution result.

11. In the **Min Execution Polling Interval** field, type the minimum value of the polling interval in milliseconds.

12. In the **Execution Polling Interval Multiplier** field, type the multiplier by which the connector increases the amount of time between polls.

13. In the **Non Proxy Host** field, type a list of hosts, separated by a comma (`,`), that the connector can access without connecting through the proxy server, when a proxy connection is enabled.

14. In the **Result Reuse Max Time** field, type the maximum age of previous query results that is considered for reuse.

15. To enable the connector to return SQL_WVARCHAR instead of SQL_VARCHAR for ARRAY, MAP, STRING, STRUCT, and VARCHAR columns, select the **Use SQL Unicode Types** check box.

16. To reuse the results of previously run queries, select the **Enable Result Reuse** check box.

17. To enable the use of the AWS result set streaming API, select the **Use Result Set Streaming** check box.

18. To save your settings and close the Advanced Options dialog box, click **OK**.

Configuring SSL Verification in Windows

If you are connecting to a Athena server that has Secure Sockets Layer (SSL) enabled, you can configure the connector to connect to an SSL-enabled socket. When using SSL to connect to a server, the connector can be configured to verify the identity of the server.

To configure SSL verification in Windows:

1. To access SSL options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **SSL Options**.

2. Select the **Verify SSL** check box.

3. To allow the common name of a CA-issued SSL certificate to not match the host name of the Athenaserver, select the **Allow Common Name Host Name Mismatch** check box.

4. To specify the CA certificates that you want to use to verify the server, do one of the following:

   - To verify the server using the trusted CA certificates from a specific `.pem` file, specify the full path to the file in the **Trusted Certificates** field and clear the **Use System Trust Store** check box.

- Or, to use the trusted CA certificates `.pem` file that is installed with the connector , leave the **Trusted Certificates** field, and clear the **Use System Trust Store** check box.

- Or, to use the Windows trust store, select the **Use System Trust Store** check box.

5. To save your settings and close the SSL Options dialog box, click **OK**.

Configuring Endpoint Override Options in Windows

You can configure endpoint override options to modify the behavior of the connector.

To configure endpoint override options in Windows:

1. To access endpoint override options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **EndPointOverride Options**.

2. In the **Streaming Endpoint Override** field, type the endpoint for the Athena streaming service corresponding to the specified Athena instance.

3. In the **Endpoint Override** field, type the endpoint for the Athena instance the connector connects to if not using the default endpoint.

4. In the **STS Endpoint Override** field, type the endpoint used for the STS service when using the assumeRoleWithSaml API to retrieve temporary credentials.

5. In the **Glue Endpoint Override** field, type the endpoint used for making AWS glue API calls.

6. To save your settings and close the EndPointOverride Options dialog box, click **OK**.

Configuring Proxy Connections in Windows

You can configure the connector to connect through a proxy server instead of connecting directly to the Athena service.

> ⚠️ **Important:**
> If you are connecting to Athena through a proxy server, make sure that the proxy server does not block port 444. The result set streaming API uses port 444 on the Athena server for outbound communications. For more information, see Use Result Set Streaming.

To configure a proxy connection in Windows:

1. To access proxy options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Proxy Options**.

2. To enable proxy connections, select the **Use Proxy** check box.

3. In the **Proxy Host** field, type the IP address or host name of your proxy server.

4. In the **Proxy Port** field, type the number of the TCP port that the proxy server uses to listen for client connections.

5. In the **Proxy Username** field, type your user name for accessing the proxy server.

6. In the **Proxy Password** field, type your password for accessing the proxy server.

7. To encrypt your credentials, click **Password Options** and then select one of the following:

   - If the credentials are used only by the current Windows user, select **Current User Only**.

   - Or, if the credentials are used by all users on the current Windows machine, select **All Users Of This Machine**.

8. To save your settings and close the HTTP Proxy Options dialog box, click **OK**.

   If the proxy server is configured to intercept SSL-encrypted connections, then in addition to specifying the proxy server information described above, you must also import the proxy server's root certificate into the Windows trust store.

   To import the proxy server's root certificate to the Windows trust store:
   a. Export the proxy server's root certificate file. You can do this using OpenSSL.

      For example, the following command exports the root certificate, originally a `.pem` file, to a `.crt` file:

      openssl x509 -outform der -in clientPublicKey.pem -out clientPublicKey.crt

      For more information, see "OpenSSL Commands" in the OpenSSL documentation: https://www.openssl.org/docs/manmaster/man1/.
   b. Import the certificate into the Window trust store. For detailed instructions, see "Installing a Certificate in the Trusted Root Certification Authorities Store" in the Microsoft Windows documentation: https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/how-to-create-temporary-certificates-for-use-during-development#to-install-a-self-signed-certificate-in-the-trusted-root-certification-authorities.

Configuring FIPS in Windows

You can configure the FIPS (Federal Information Processing Standards) module to ensure the security, quality, and processing compatibility of various services.

> ℹ **Note:** To enable FIPS support, a separate FIPS downloadable package is available for assistance.

To configure FIPS in Windows:

1. Unzip the OpenSSL_3.0_Modules_Windows_vs2022.zip Windows package released with the connector.

2. Follow the instructions mentioned in the **README.md** file.

   > ℹ **Note:** Make sure that all the FIPS module binary files are present in the OPENSSL_ MODULES path, otherwise the connector does not work as expected.

Exporting a Data Source Name in Windows

After you configure a DSN, you can export it to be used on other machines. When you export a DSN, all of its configuration settings are saved in a `.sdc` file. You can then distribute the `.sdc` file to other users so that they can import your DSN configuration and use it on their machines.

To export a Data Source Name in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, click **Configure**, and then click **Logging Options**.

2. Click **Export Configuration**, specify a name and location for the exported DSN, and then click **Save**.

   Your DSN is saved as a `.sdc` file in the location that you specified.

Importing a Data Source Name in Windows

You can import a DSN configuration from a `.sdc` file and then use those settings to connect to your data source.

To import a Data Source Name in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, select the DSN, click **Configure**, and then click **Logging Options**.

2. Click **Import Configuration**, browse to select the `.sdc` file that you want to import the DSN configuration from, and then click **Open**.

3. Click **OK** to close the Logging Options dialog box.

   The SimbaAthena ODBC Driver DSN Setup dialog box loads the configuration settings from the selected `.sdc` file. You can now save this DSN and use it to connect to your data source.

Configuring Logging Options in Windows

To help troubleshoot issues, you can enable logging. In addition to functionality provided in the Simba Simba Amazon Athena ODBC Connector, the ODBC Data Source Administrator provides tracing functionality.

> ⚠️ **Important:** Only enable logging or tracing long enough to capture an issue. Logging or tracing decreases performance and can consume a large quantity of disk space.

Configuring Connector-wide Logging Options

The settings for logging apply to every connection that uses the Simba Amazon Athena ODBC Connector, so make sure to disable the feature after you are done using it. To configure logging for the current connection, see Configuring Logging for the Current Connection.

To enable connector-wide logging in Windows:

1. To access logging options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Logging Options**.

2. From the **Log Level** drop-down list, select the logging level corresponding to the amount of information that you want to include in log files:

Logging Level	Description
OFF	Disables all logging.
FATAL	Logs severe error events that lead the connector to abort.
ERROR	Logs error events that might allow the connector to continue running.
WARNING	Logs events that might result in an error if action is not taken.
INFO	Logs general information that describes the progress of the connector.
DEBUG	Logs detailed information that is useful for debugging the connector.
TRACE	Logs all connector activity.

> **Note:** If the Use AWS Logger check box is selected, the connector also logs information from AWS API calls.

3. In the **Log Path** field, specify the full path to the folder where you want to save log files. You can type the path into the field, or click **Browse** and then browse to select the folder.

4. In the **Max Number Files** field, type the maximum number of log files to keep.

> **Note:** After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

5. In the **Max File Size** field, type the maximum size of each log file in megabytes (MB).

> **Note:** After the maximum file size is reached, the connector creates a new file and continues logging.

6. To log information about AWS API calls, select the **Use Aws Logger** check box.

7. Click **OK**.

8. Restart your ODBC application to make sure that the new settings take effect.

The Simba Amazon Athena ODBC Connector produces the following log files at the location you specify in the Log Path field:

- A `simbaathenaodbcdriver.log` file that logs connector activity that is not specific to a connection.

- A `simbaathenaodbcdriver_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 you enable the `UseLogPrefix` connection property, the connector prefixes the log file name with the user name associated with the connection and the process ID of the application through which the connection is made. For more information, see UseLogPrefix.

To disable connector logging in Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Logging Options**.

2. From the **Log Level** drop-down list, select **LOG_OFF**.

3. Click **OK**.

4. Restart your ODBC application to make sure that the new settings take effect.

Configuring Logging for the Current Connection

You can configure logging for the current connection by setting the logging configuration properties in the DSN or in a connection string. For information about the logging configuration properties, see Configuring Logging Options in Windows. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

> **Note:** If the LogLevel configuration property is passed in via the connection string or DSN, the rest of the logging configurations are read from the connection string or DSN and not from the existing connector-wide logging configuration.

To configure logging properties in the DSN, you must modify the Windows registry. For information about the Windows registry, see the Microsoft Windows documentation.

> **Important:** Editing the Windows Registry incorrectly can potentially cause serious, system-wide problems that may require re-installing Windows to correct.

To add logging configurations to a DSN in Windows:

1. Choose one:

   - If you are using Windows 7 or earlier, click **Start**, then type **regedit** in the **Search** field, and then click **regedit.exe** in the search results.

   - Or, if you are using Windows 8 or later, on the Start screen, type **regedit**, and then click the **regedit** search result.

2. Navigate to the appropriate registry key for the bitness of your connector and your machine:

   - 32-bit System DSNs: **HKEY_LOCAL_ MACHINE\SOFTWARE\WOW6432Node\ODBC\ODBC.INI***[DSN Name]*

   - 64-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI***[DSN Name]*

   - 32-bit and 64-bit User DSNs: **HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI\ *[DSN Name]***

3. For each configuration option that you want to configure for the current connection, create a value by doing the following:

    a. If the key name value does not already exist, create it. Right-click the *[DSN Name]* and then select **New > String Value**, type the key name of the configuration option, and then press **Enter**.

    b. Right-click the key name and then click **Modify**.

       To confirm the key names for each configuration option, see Connector Configuration Options.

    c. In the Edit String dialog box, in the **Value Data** field, type the value for the configuration option.

4. Close the Registry Editor.

5. Restart your ODBC application to make sure that the new settings take effect.

Verifying the Connector Version Number in Windows

If you need to verify the version of the Simba Amazon Athena ODBC Connector that is installed on your Windows machine, you can find the version number in the ODBC Data Source Administrator.

To verify the connector version number in Windows:

1. From the Start menu, go to **ODBC Data Sources**.

> **ⓘ** **Note:** Make sure to select the ODBC Data Source Administrator that has the same bitness as the client application that you are using to connect to Athena.

2. Click the **Drivers** tab and then find the Simba Amazon Athena ODBC Connector in the list of ODBC Connectors that are installed on your system. The version number is displayed in the **Version** column.

macOS Connector

This section provides an overview of the Connector in the mac OS platform, outlining the required system specifications and the steps for installing and configuring the connector in mac OS environments.

macOS System Requirements

Install the connector on client machines where the application is installed. Each client machine that you install the connector on must meet the following minimum system requirements:

- One of the following macOS versions:
  - macOS 11 (Universal Binary – Intel and ARM support)
  - macOS 12 (Universal Binary – Intel and ARM support)

- 150MB of available disk space

- One of the following ODBC driver managers installed:
  - iODBC 3.52.9 or later
  - unixODBC 2.3.6 or later

Installing the Connector in macOS

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Connectors Installation Guide*.

The Simba Amazon Athena ODBC Connector is available for macOS as a `.dmg` file named `Simba Athena 1.3.dmg`. The connector supports both 32- and 64-bit client applications.

To install the Simba Amazon Athena ODBC Connector in macOS:

1. Double-click **Simba Athena 1.3.dmg** to mount the disk image.

2. In the installer, click **Continue**.

3. On the Software License Agreement screen, click **Continue**, and when the prompt appears, click **Agree** if you agree to the terms of the License Agreement.

4. Optionally, to change the installation location, click **Change Install Location**, then select the desired location, and then click **Continue**.

   > **Note:** By default, the connector files are installed in the `/Library/simba/athenaodbc` directory.

5. To accept the installation location and begin the installation, click **Install**.

6. When the installation completes, click **Close**.

7. If you received a license file through email, then copy the license file into the `/lib` subfolder in the connector installation directory. You must have root privileges when changing the contents of this

folder.

For example, if you installed the connector to the default location, you would copy the license file into the `/Library/simba/athenaodbc/lib` folder.

Next, configure the environment variables on your machine to make sure that the ODBC Driver manager can work with the connector. For more information, see Configuring the ODBC Driver Manager in Non-Windows Machines

Next, configure the environment variables on your machine to make sure that the ODBC Driver manager can work with the connector. For more information, Configuring the ODBC Driver Manager in Non-Windows Machines.

Configuring FIPS in mac OS

You can configure the FIPS (Federal Information Processing Standards) module to ensure the security, quality, and processing compatibility of various services.

> ℹ️ **Note:** To enable FIPS support, a separate FIPS downloadable package is available for assistance.

To configure FIPS in mac OS:

1. Unzip the OpenSSL_3.0_Modules_OSX_ARM_xcode13_2.tar.gz package released with the connector.

2. Follow the instructions mentioned in the **README.md** file.

   > ℹ️ **Note:** Ensure that all the FIPS module binary files are present in the OPENSSL_ MODULES path, otherwise the connector does not work as expected.

Verifying the Connector Version Number in macOS

If you need to verify the version of the Simba Amazon Athena ODBC Connector that is installed on your macOS machine, you can query the version number through the Terminal.

To verify the connector version number in macOS:

- At the Terminal, run the command:

```
pkgutil --info com.simba.athenaodbc
```

The command returns information about the Simba Amazon Athena ODBC Connector that is installed on your machine, including the version number.

Linux Connector

This section provides an overview of the Connector in the Linux platform, outlining the required system specifications and the steps for installing and configuring the connector in Linux environments.

Linux System Requirements

Install the connector on client machines where the application is installed. Each client machine that you install the connector on must meet the following minimum system requirements:

- One of the following distributions:
  - Red Hat® Enterprise Linux® (RHEL) 8, 9, or 10
  - SUSE Linux Enterprise Server (SLES) 12 or 15
  - Debian 11, 12, or 13
  - Ubuntu 22.04 or 24.04

- 150MB of available disk space

- One of the following ODBC driver managers installed:
  - iODBC 3.52.9 or later
  - unixODBC 2.2.14 or later

To install the connector, you must have root access on the machine.

If you are using the RPM file to install the connector on Debian or Ubuntu, you must also have the `alien` utility installed. The `alien` utility is available on SourceForge: https://sourceforge.net/projects/alien-pkg-convert/.

Installing the Connector Using the RPM File

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Connectors Installation Guide*.

On 64-bit editions of Linux, you can execute both 32- and 64-bit applications. However, 64-bit applications must use 64-bit connectors, and 32-bit applications must use 32-bit connectors. Make sure that you use a connector whose bitness matches the bitness of the client application:

- `simbaathena-[Version]-[Release].i686.rpm` for the 32-bit connector

- `simbaathena-[Version]-[Release].x86_64.rpm` for the 64-bit connector

The placeholders in the file names are defined as follows:

- *[Version]* is the version number of the connector.

- *[Release]* is the release number for this version of the connector.

To install the Simba Amazon Athena ODBC Connector using the RPM File:

1. Log in as the root user.

2. If you are installing the connector on a Debian or Ubuntu machine, download and install the `alien` utility:
   a. Download the package from SourceForge: https://sourceforge.net/projects/alien-pkg-convert/.
   b. From the command line, run the following command:

      sudo apt-get install alien

3. Navigate to the folder containing the RPM package for the connector.

4. Depending on the Linux distribution that you are using, run one of the following commands from the command line, where *[RPMFileName]* is the file name of the RPM package:

   - If you are using Red Hat Enterprise Linux, Amazon Linux, or CentOS, run the following command:

     ```
     yum --nogpgcheck localinstall [RPMFileName]
     ```

   - Or, if you are using SUSE Linux Enterprise Server, run the following command:

     ```
     zypper install [RPMFileName]
     ```

   - Or, if you are using Debian or Ubuntu, run the following command:

     alien -i *[RPMFileName]*

5. If you received a license file through email, then copy the license file into the `/opt/simba/athenaodbc/lib/32` or `/opt/simba/athenaodbc/lib/64` folder, depending on the version of the connector that you installed.

Next, configure the environment variables on your machine to make sure that the ODBC Driver manager can work with the connector. For more information, see Configuring the ODBC Driver Manager in Non-Windows Machines

Configuring FIPS in Linux

You can configure the FIPS (Federal Information Processing Standards) module to ensure the security, quality, and processing compatibility of various services.

> **Note:** To enable FIPS support, a separate FIPS downloadable package is available for assistance.

To configure FIPS in Linux:

1. Unzip the OpenSSL_3.0_Modules_Linux_gcc5_5.tar.gz Linux package released with the connector.

2. Follow the instructions mentioned in the **README.md** file.

> ℹ **Note:** Make sure that all the FIPS module binary files are present in the OPENSSL_ MODULES path, otherwise the connector does not work as expected.

Verifying the Connector Version Number in Linux

If you need to verify the version of the Simba Amazon Athena ODBC Connector that is installed on your Linux machine, you can query the version number through the command-line interface if the connector was installed using an RPM file. Alternatively, you can search the connector's binary file for version number information.

To verify the connector version number in Linux using the command-line interface:

- Depending on your package manager, at the command prompt, run one of the following commands:

    - yum list | grep AthenaODBC

    - rpm -qa | grep AthenaODBC

    - dpkg -l | grep simbaathenaodbc

The command returns information about the Simba Amazon Athena ODBC Connector that is installed on your machine, including the version number.

To verify the connector version number in Linux using the binary file:

1. Navigate to the `/lib` subfolder in your connector installation directory. By default, the path to this directory is: `/opt/simba/athenaodbc/lib`.

2. Open the connector's `.so` binary file in a text editor, and search for the text `$driver_version_ sb$:`. The connector's version number is listed after this text.

Configuring the ODBC Driver Manager in Non-Windows Machines

To make sure that the ODBC Driver manager on your machine is configured to work with the Simba Amazon Athena ODBC Connector, do the following:

- Set the library path environment variable to make sure that your machine uses the correct ODBC Driver manager. For more information, see Specifying ODBC Driver Managers in Non-Windows Machines .

- If the connector configuration files are not stored in the default locations expected by the ODBC driver manager, then set environment variables to make sure that the Driver manager locates and uses those files. For more information, see Specifying the Locations of the Connector Configuration Files.

  After configuring the ODBC Driver manager, you can configure a connection and access your data store through the connector.

Specifying ODBC Driver Managers in Non-Windows Machines

You need to make sure that your machine uses the correct ODBC Driver manager to load the connector. To do this, set the library path environment variable.

macOS

If you are using a macOS machine, then set the DYLD_LIBRARY_PATH environment variable to include the paths to the ODBC driver manager libraries. For example, if the libraries are installed in `/usr/local/lib`, then run the following command to set DYLD_LIBRARY_PATH for the current user session:

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:/usr/local/lib
```

For information about setting an environment variable permanently, refer to the macOS shell documentation.

Linux

If you are using a Linux machine, then set the LD_LIBRARY_PATH environment variable to include the paths to the ODBC driver manager libraries. For example, if the libraries are installed in `/usr/local/lib`, then run the following command to set LD_LIBRARY_PATH for the current user session:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

For information about setting an environment variable permanently, refer to the Linux shell documentation.

Specifying the Locations of the Connector Configuration Files

By default, ODBC Driver managers are configured to use hidden versions of the `odbc.ini` and `odbcinst.ini` configuration files (named `.odbc.ini` and `.odbcinst.ini`) located in the home directory, as well as the `simba.athenaodbc.ini` file in the `lib` subfolder of the connector installation directory. If you store these configuration files elsewhere, then you must set the environment variables described below so that the driver manager can locate the files.

If you are using iODBC, do the following:

- Set ODBCINI to the full path and file name of the `odbc.ini` file.

- Set ODBCINSTINI to the full path and file name of the `odbcinst.ini` file.

- Set SIMBA_ATHENA_ODBC_INI to the full path and file name of the `simba.athenaodbc.ini` file.

If you are using unixODBC, do the following:

- Set ODBCINI to the full path and file name of the `odbc.ini` file.

- Set ODBCSYSINI to the full path of the directory that contains the `odbcinst.ini` file.

- Set SIMBA_ATHENA_ODBC_INI to the full path and file name of the `simba.athenaodbc.ini` file.

For example, if your `odbc.ini` and `odbcinst.ini` files are located in `/usr/local/odbc` and your `simba.athenaodbc.ini` file is located in `/etc`, then set the environment variables as follows:

For iODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini

export ODBCINSTINI=/usr/local/odbc/odbcinst.ini

export SIMBA_ATHENA_ODBC_INI=/etc/simba.athenaodbc.ini
```

For unixODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini

export ODBCSYSINI=/usr/local/odbc

export SIMBA_ATHENA_ODBC_INI=/etc/simba.athenaodbc.ini
```

To locate the `simba.athenaodbc.ini` file, the connector uses the following search order:

1. If the SIMBA_ATHENA_ODBC_INI environment variable is defined, then the connector searches for the file specified by the environment variable.

2. The connector searches the directory that contains the connector library files for a file named `simba.athenaodbc.ini`.

3. The connector searches the current working directory of the application for a file named `simba.athenaodbc.ini`.

4. The connector searches the home directory for a hidden file named `simba.athenaodbc.ini` (prefixed with a period).

5. The connector searches the `/etc` directory for a file named `simba.athenaodbc.ini`.

Configuring ODBC Connections in Non-Windows Machine

The following sections describe how to configure ODBC connections when using the Simba Amazon Athena ODBC Connector on non-Windows platforms:

- Creating a Data Source Name on a Non-Windows Machine

- Configuring a DSN-less Connection in a Non-Windows Machine

- Configuring Authentication on Non-Windows Machines

- Configuring Proxy Connections on Non-Windows Machines

- Configuring Query Result Encryption on a Non-Windows Machine

- Configuring Logging Options in a Non-Windows Machine

- Testing the Connection in Non-Windows Machine

Creating a Data Source Name on a Non-Windows Machine

When connecting to your data store using a DSN, you only need to configure the `odbc.ini` file. Set the properties in the `odbc.ini` file to create a DSN that specifies the connection information for your data store. For information about configuring a DSN-less connection instead, see Configuring a DSN-less Connection in a Non-Windows Machine.

If your machine is already configured to use an existing `odbc.ini` file, then update that file by adding the settings described below. Otherwise, copy the `odbc.ini` file from the `Setup` subfolder in the connector installation directory to the home directory, and then update the file as described below.

To create a Data Source Name on a non-Windows machine:

1. In a text editor, open the `odbc.ini` configuration file.

   > **Note:** If you are using a hidden copy of the `odbc.ini` file, you can remove the period (`.`) from the start of the file name to make the file visible while you are editing it.

2. In the `[ODBC Data Sources]` section, add a new entry by typing a name for the DSN, an equal sign (=), and then the name of the connector.

For example, on a macOS machine:

[ODBC Data Sources]

Sample DSN=Simba Amazon Athena ODBC Connector

As another example, for a 32-bit connector on a Linux machine:

[ODBC Data Sources]

Sample DSN=Simba Amazon Athena ODBC Connector 32-bit

3. Create a section that has the same name as your DSN, and then specify configuration options as key-value pairs in the section:

   a. Set the `Driver` property to the full path of the connector library file that matches the bitness of the application.

   For example, on a macOS machine:

   Driver=/Library/simba/athenaodbc/lib/libathenaodbc_sbu.dylib

   As another example, for a 32-bit connector on a Linux machine:

   Driver=/opt/simba/athenaodbc/lib/32/libathenaodbc_sb32.so

   b. Set the `AwsRegion` property to the AWS region of the Athena instance that you want to connect to.

   For example:

   AwsRegion=us-east-2

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

   c. Set the `S3OutputLocation` property to the path of the Amazon S3 location where you want to store query results, prefixed by `s3://`.

   For example, to store results in a folder named "test-folder-1" inside an S3 bucket named "query-results-bucket", you would specify the following:

   S3OutputLocation=s3://query-results-bucket/test-folder-1

   d. Configure authentication by specifying the authentication mechanism to use and providing your credentials. For more information, see Configuring Authentication on Non-Windows Machines.

   e. Optionally, configure the connector to connect to Athena through a proxy server. For more information, see Configuring Proxy Connections on Non-Windows Machines.

   f. Optionally, configure encryption for your query results. For more information, see Configuring Query Result Encryption on a Non-Windows Machine.

   g. Optionally, set additional key-value pairs as needed to specify other optional connection settings. For detailed information about all the configuration options supported by the Simba Amazon Athena ODBC Connector, see Connector Configuration Properties.

4. Save the `odbc.ini` configuration file.

   > **Note:** If you are storing this file in its default location in the home directory, then prefix the file name with a period (`.`) so that the file becomes hidden. If you are storing this file in another location, then save it as a non-hidden file (without the prefix), and make sure that the ODBCINI environment variable specifies the location. For more information, see Specifying the Locations of the Connector Configuration Files.

For example, the following is an `odbc.ini` configuration file for macOS containing a DSN that connects to Athena using IAM credentials:

[ODBC Data Sources]

Sample DSN=Simba Amazon Athena ODBC Connector

[Sample DSN]

Driver=/Library/simba/athenaodbc/lib/libathenaodbc_sbu.dylib

AuthenticationType=IAM Credentials

UID=ABCABCABC123ABCABC45

PWD=bCD+E1f2Gxhi3J4klmN/OP5QrSTuvwXYzabcdEF

AwsRegion=us-east-2

S3OutputLocation=s3://simba-athena-results/

As another example, the following is an `odbc.ini` configuration file for a 32-bit connector on a Linux machine, containing a DSN that connects to Athena using IAM credentials:

[ODBC Data Sources]

Sample DSN=Simba Amazon Athena ODBC Connector 32-bit

[Sample DSN]

Driver=/opt/simba/athenaodbc/lib/32/libathenaodbc_sb32.so

AuthenticationType=IAM Credentials

UID=ABCABCABC123ABCABC45

PWD=bCD+E1f2Gxhi3J4klmN/OP5QrSTuvwXYzabcdEF

AwsRegion=us-east-2

S3OutputLocation=s3://simba-athena-results/

You can now use the DSN in an application to connect to the data store.

Configuring a DSN-less Connection in a Non-Windows Machine

To connect to your data store through a DSN-less connection, you need to define the connector in the `odbcinst.ini` file and then provide a DSN-less connection string in your application.

If your machine is already configured to use an existing `odbcinst.ini` file, then update that file by adding the settings described below. Otherwise, copy the `odbcinst.ini` file from the `Setup` subfolder in the connector installation directory to the home directory, and then update the file as described below.

To define a connector on a non-Windows machine:

1. In a text editor, open the `odbcinst.ini` configuration file.

   Note: If you are using a hidden copy of the `odbcinst.ini` file, you can remove the period (`.`) from the start of the file name to make the file visible while you are editing it.

2. In the `[ODBC Drivers ]` section, add a new entry by typing a name for the connector, an equal sign (=), and then `Installed`.

For example:

[ODBC Drivers]

Simba Amazon Athena ODBC Connector=Installed

3. Create a section that has the same name as the connector (as specified in the previous step), and then specify the following configuration options as key-value pairs in the section:
   a. Set the `Driver` property to the full path of the connector library file that matches the bitness of the application.

      For example, on a macOS machine:

      Driver=/Library/simba/athenaodbc/lib/libathenaodbc_sbu.dylib

      As another example, for a 32-bit connector on a Linux machine:

      Driver=/opt/simba/athenaodbc/lib/32/libathenaodbc_sb32.so
   b. Optionally, set the `Description` property to a description of the connector.

      For example:

      Description=Simba Amazon Athena ODBC Connector

4. Save the `odbcinst.ini` configuration file.

   > **Note:** If you are storing this file in its default location in the home directory, then prefix the file name with a period (`.`) so that the file becomes hidden. If you are storing this file in another location, then save it as a non-hidden file (without the prefix), and make sure that the ODBCINSTINI or ODBCSYSINI environment variable specifies the location. For more information, see Specifying the Locations of the Connector Configuration Files.

For example, the following is an `odbcinst.ini` configuration file for macOS:

[ODBC Drivers]

Simba Amazon Athena ODBC Connector=Installed

[Simba Amazon Athena ODBC Connector]

Description=Simba Amazon Athena ODBC Connector

Driver=/Library/simba/athenaodbc/lib/libathenaodbc_sbu.dylib

As another example, the following is an `odbcinst.ini` configuration file for both the 32- and 64-bit connectors in Linux:

[ODBC Drivers]

Simba Amazon Athena ODBC Connector 32-bit=Installed

Simba Amazon Athena ODBC Connector 64-bit=Installed

[Simba Amazon Athena ODBC Connector 32-bit]

Description=Simba Amazon Athena ODBC Connector (32-bit)

Driver=/opt/simba/athenaodbc/lib/32/libathenaodbc_sb32.so

[Simba Amazon Athena ODBC Connector 64-bit]

Description=Simba Amazon Athena ODBC Connector (64-bit)

Driver=/opt/simba/athenaodbc/lib/64/libathenaodbc_sb64.so

You can now connect to your data store by providing your application with a connection string where the `Driver` property is set to the connector name specified in the `odbcinst.ini` file, and all the other necessary connection properties are also set. For more information, see "DSN-less Connection String Examples" in Using a Connection String.

Configuring Authentication on Non-Windows Machines

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

- Using the Default Credentials Provider Chain on Non-Windows Machines

- Using IAM Credentials on Non-Windows Machines

- Using an IAM Profile on Non-Windows Machines

- Using an Instance Profile on Non-Windows Machines

- Using the Active Directory Federation Services (AD FS) Credentials Provider on a Non-Windows Machine

- Using the Azure AD Credentials Provider on a Non-Windows Machine

- Using the Browser Azure AD Credentials Provider on a Non-Windows Machine

- Using a Browser Plugin for a SAML Service on a Non-Windows Machine

- Using a JSON Web Token (JWT)

- Using the Okta Service on a non-Windows machine

- Using Okta MFA with Okta Verify on a non-Windows machine

- Using Okta MFA with Google Authenticator on a non-Windows machine

- Using Okta MFA with SMS Authentication on a non-Windows machine

- Using PingFederate Service

- Using an External Credentials Service

You can set the connection properties described below in a connection string or in a DSN (in the `odbc.ini` file). Settings in the connection string take precedence over settings in the DSN.

Using the Default Credentials Provider Chain on Non-Windows Machines

You can configure the connector to authenticate the connection using credentials that are stored in one of the locations in the default credentials provider chain. The connector looks for a valid access key and secret key pair by checking the following locations, in the following order:

1. The AWS credentials file stored in the `~/.aws/credentials` directory.

2. The AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY system environment variables.

3. The instance profile from the Amazon EC2 Instance Metadata Service.

For detailed information about configuring default credentials, see "Providing AWS Credentials" in the *AWS SDK for C++ Developer Guide*: http://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/credentials.html.

To configure authentication using the default credentials provider chain on a non-Windows machine:

- Set the `AuthenticationType` property to `Default Credentials`.

Using IAM Credentials on Non-Windows Machines

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

If you are using temporary credentials, which are only valid for a limited amount of time, then you must also provide a session token. For more information, see "Temporary Security Credentials" in the *AWS Identity and Access Management User Guide*: http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html.

To configure authentication using IAM credentials on a non-Windows machine:

1. Set the `AuthenticationType` property to `IAM Credentials`.

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

3. Set the `PWD` property to the secret key provided by your AWS account.

4. If you are using temporary credentials, set the `SessionToken` property to the session token generated by the AWS Security Token Service.

5. Optionally, to retrieve SAML credentials using Lake Formation service, set the `LakeformationEnabled` to true.

Using an IAM Profile on Non-Windows Machines

You can configure the connector to authenticate the connection using credentials that are associated with an IAM profile in a credentials file.

By default, the connector uses the credentials associated with a profile named `default` in the credentials file found in the `~/.aws/credentials` directory. To use a different profile, specify the profile name in your connection settings. To use a different credentials file, set the AWS_SHARED_CREDENTIALS_FILE system environment variable to the full path of your credentials file.

For information about the format of a credentials file, see the "AWS Credentials File Format" section from the "Working with AWS Credentials" page in the *AWS SDK for Java Developer Guide*: http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html.

To configure authentication using an IAM profile on a non-Windows machine:

1. Set the `AuthenticationType` property to `IAM Profile`.

2. Set the `AWSProfile` property to the name of the profile to use.

3. Optionally, set the `PreferredRole` property to the Amazon Resource Name (ARN) of the role you want to assume when authenticating through an external credential service.

Using an Instance Profile on Non-Windows Machines

> **Note:** Because Amazon EC2 instances are not available for macOS at this time, the macOS version of the Simba Amazon Athena ODBC Connector cannot use this authentication method.

You can configure the connector to authenticate the connection using credentials that have been loaded from the Amazon EC2 Instance Metadata Service into an instance profile.

Instance profiles contain authorization information such as roles, permissions, and credentials, and are automatically created by Amazon EC2 for each IAM role that is defined for an EC2 instance. For more information, see "IAM Roles for Amazon EC2" in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html.

To configure authentication using an instance profile on a non-Windows machine:

- Set the `AuthenticationType` property to `Instance Profile`.

Using the Active Directory Federation Services (AD FS) Credentials Provider on a Non-Windows Machine

You can configure the connector to authenticate the connection using credentials obtained from the AD FS credentials provider. To do this, you must specify information about the AD FS service, such as the host and port of the server where the service is hosted.

To configure authentication using AD FS on a non-Windows machine:

1. Set the `AuthenticationType` property to `ADFS`.

2. To specify your credentials for accessing the AD FS server, do the following:
   a. Set the `UID` property to the user name that you use to access the AD FS server. You can include the domain name using the format *[DomainName]\[UserName]*.
   b. Set the `PWD` property to the password corresponding to the user name that you provided in the previous step.

3. To specify AD FS service information, do the following:
   a. Set the `IdP_Host` property to the host name of the AD FS service.

   > **!** **Important:**
   > The host name cannot include any slashes (`/`).

   b. Optionally, set the `IdP_Port` property to the number of the port that the AD FS service host uses to listen for requests.

   > **i** **Note:** The exact port number that you need to specify may differ depending on the AD FS server configuration. If you are not sure which port to specify, contact your system administrator.

4. Optionally, set the `Preferred_Role` property to the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through AD FS.

5. Optionally, set the `Duration` property to the duration, in seconds, of the role session.

6. If the AD FS service must be accessed through an HTTP proxy, set the `UseProxyForIdP` property to `1`. For information about configuring the proxy connection, see Configuring Proxy Connections on Non-Windows Machines.

7. Optionally, if you do not want the connector to verify the AD FS server certificate, set the `SSL_Insecure` property to `false`.

Using the Azure AD Credentials Provider on a Non-Windows Machine

You can configure the connector to authenticate the connection using credentials obtained from the Azure AD credentials provider. To do this, you must specify information about the Azure AD service, such as the Client ID and Secret and the Tenant ID.

To configure authentication using Azure AD on a non-Windows machine:

1. Set the `AuthenticationType` property to `AzureAD`.

2. To specify your credentials for accessing the Azure AD server, do the following:
   a. Set the `UID` property to the user name that you use to access the Azure AD server.
   b. Set the `PWD` property to the password corresponding to the user name that you provided in the previous step.

3. Optionally, set the `Preferred_Role` property to the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Azure AD.

4. Optionally, set the `Duration` property to the duration, in seconds, of the role session.

5. Set the `idp_tenant` property to the Azure AD-provided unique ID associated with your Athena application.

6. Set the `client_id` property to the Client ID to use when authenticating the connection using the Azure AD service.

7. Set the `client_secret` property to the Client Secret to use when authenticating the connection using the Azure AD service.

Using the Browser Azure AD Credentials Provider on a Non-Windows Machine

You can configure the connector to authenticate the connection using credentials obtained from the Azure AD credentials provider. To do this, you must specify information about the Azure AD service, such as the Client ID and Secret and the Tenant ID.

To configure authentication using Azure AD on a non-Windows machine:

1. Set the `AuthenticationType` property to `BrowserAzureAD`.

2. To specify your credentials for accessing the Azure AD server, do the following:

   a. Set the `UID` property to the user name that you use to access the Azure AD server.

   b. Set the `PWD` property to the password corresponding to the user name that you provided in the previous step.

3. Optionally, set the `Preferred_Role` property to the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Azure AD.

4. Optionally, set the `Duration` property to the duration, in seconds, of the role session.

5. Set the `idp_tenant` property to the Azure AD-provided unique ID associated with your Athena application.

6. Set the `client_id` property to the Client ID to use when authenticating the connection using the Azure AD service.

7. Optionally, set the `client_secret` property to the Client Secret to use when authenticating the connection using the Azure AD service.

8. Optionally, set the `timeout` property to the maximum amount of time, in seconds, that the connector is to wait for the redirect URI to fetch the authorization code during Browser Azure AD authentication.

   > **Note:** The redirect URI must be in the following format:
   > `http://localhost:portnumber/athena`.

Using a Browser Plugin for a SAML Service on a Non-Windows Machine

You can configure the connector to use a browser plugin to authenticate your connection through a SAML service such as Okta, Ping, or AD FS.

To configure IAM authentication using a browser plugin on a non-Windows machine:

1. Set the `AuthenticationType` property to `BrowserSAML`.

2. To specify your credentials for accessing the Azure AD server, do the following:

    a. Set the `UID` property to the user name that you use to access the Azure AD server.

    b. Set the `PWD` property to the password corresponding to the user name that you provided in the previous step.

3. Optionally, set the `Preferred_Role` property to the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Azure AD.

4. Optionally, set the `Duration` property to the duration, in seconds, of the role session.

5. Set the `Login_URL` property to the URL for the resource on the identity provider's website.

6. Optionally, set the `Listen_Port` property to the number of the port that the connector uses to receive the SAML response from the identity provider.

7. Optionally, set the `timeout` property to the maximum amount of time, in seconds, that the connector is to wait for the redirect URI to fetch the authorization code during Browser Azure AD authentication.

Using a JSON Web Token (JWT)

You can configure the connector to authenticate your connection by using a token obtained from the web identity provider.

To configure IAM authentication using a JWT on a non-Windows machine:

1. Set the `AuthenticationType` property to `JWT`.

2. Set the `web_token` property to the JSON web token generated after OAuth authentication from Azure Active directory.

3. Optionally, set the `role_session_name` property to the name of the assumed role session.

Using the Okta Service on a non-Windows machine

You can configure the connector to authenticate the connection using credentials obtained from the Okta credentials provider. To do this, you must specify information about the Okta service, such as the host name of the Okta service and the Okta application ID.

To configure authentication using Okta on a non-Windows machine:

1. Set the `AuthenticationType` property to `Okta`.

2. Set the `UID` property to the user name associated with your Okta account.

3. Set the `PWD` property to the password associated with your Okta user name.

4. Set the `IdP_Host` property to the host name of the Okta service.

5. Set the `App_ID` property to the Okta-supplied ID associated with your Athena application.

6. Optionally, set the `Preferred_Role` property to the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.

7. Optionally, set the `App_Name` property to the name of your Okta application.

Using Okta MFA with Okta Verify on a non-Windows machine

You can configure the connector to authenticate the connection using credentials obtained from Okta MFA with Okta Verify factor.

> ! **Important:**
> Ensure that the Okta Verify factor is enabled on Okta.

To enroll with Okta Verify with TOTP:

1. Scan the QR code displayed on the browser using the Okta Verify app.

2. In the dialog box, enter the password from the app

3. Activate the device.

To enroll with Okta Verify with push:

1. Scan the QR code displayed on the browser using the Okta Verify app.

2. Approve the push notification sent to your device.

To configure authentication using Okta Verify factor on a non-Windows machine:

1. Set the `AuthenticationType` property to `Okta`.

2. Set the `UID` property to the user name associated with your Okta account.

3. Set the `PWD` property to the password associated with your Okta user name.

4. Set the `IdP_Host` property to the host name of the Okta service.

5. Set the `App_ID` property to the Okta-supplied ID associated with your Athena application.

6. Optionally, set the `okta_mfa_wait_time` property to the MFA timeout value, in seconds.

7. Set the `okta_mfa_type` property to either `oktaverifywithtotp` or `oktaverifywithpush`.

8. Optionally, set the `Preferred_Role` property to the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.

9. Optionally, set the `App_Name` property to the name of your Okta application.

Using Okta MFA with Google Authenticator on a non-Windows machine

You can configure the connector to authenticate the connection using credentials obtained from Okta MFA with Google Authenticator.

> ! **Important:**
> Ensure that the Google Authenticator factor is enabled on Okta.

To enroll with Okta Google Authenticator:

1. Scan the QR code displayed on the browser using the Okta Verify app.

2. In the dialog box, type the password from the app.

3. Activate the device.

To configure authentication using Okta MFA with Google Authenticator on a non-Windows machine:

1. Set the `AuthenticationType` property to `Okta`.

2. Set the `UID` property to the user name associated with your Okta account.

3. Set the `PWD` property to the password associated with your Okta user name.

4. Set the `IdP_Host` property to the host name of the Okta service.

5. Set the `App_ID` property to the Okta-supplied ID associated with your Athena application.

6. Optionally, set the `okta_mfa_wait_time` property to the MFA timeout value, in seconds.

7. Set the `okta_mfa_type` property to `GoogleAuthenticator`.

8. Optionally, set the `Preferred_Role` property to the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.

9. Optionally, set the `App_Name` property to the name of your Okta application.

Using Okta MFA with SMS Authentication on a non-Windows machine

You can configure the connector to authenticate the connection using credentials obtained from Okta MFA with SMS Authentication.

> ⚠ **Important:**
> Ensure that the SMS Authentication factor is enabled on Okta.

To enroll with Okta SMS Authentication:

- Enter the password sent to your mobile device.

To configure authentication using Okta MFA with SMS Authentication on a non-Windows machine:

1. Set the `AuthenticationType` property to `Okta`.

2. Set the `UID` property to the user name associated with your Okta account.

3. Set the `PWD` property to the password associated with your Okta user name.

4. Set the `IdP_Host` property to the host name of the Okta service.

5. Set the `App_ID` property to the Okta-supplied ID associated with your Athena application.

6. Optionally, set the `okta_mfa_wait_time` property to the MFA timeout value, in seconds.

7. Set the `okta_mfa_type` property to `SmsAuthentication`.

8. Set the `okta_phone_number` property to a US or Canadian number.

> **Note:** Okta only supports a US or Canadian number. If a number is not preceded with the country code, +1 is automatically added.

9. Optionally, set the `Preferred_Role` property to the Amazon Resource Name (ARN) of the role that you want to assume when authenticated through Okta.

10. Optionally, set the `App_Name` property to the name of your Okta application.

Using PingFederate Service

You can configure the connector to authenticate your connection through IAM authentication using the credentials stored in the PingFederate service.

To configure IAM authentication using PingFederate service on a non-Windows machine:

1. Set the `UID` property to the user name associated with your Ping account.

2. Set the `PWD` property to the password associated with your Ping user name.

3. Set the `plugin_name` property to `Ping`.

4. Set the `IdP_Host` property to the address of the service host.

5. Set the `IdP_Port` property to the port number that the service listens at.

6. Set the `Preferred_Role` property to the name or ID for the IAM Role that you want the user to assume when logged in to Athena.

7. To skip verification of the SSL certificate of the IDP server, set the `SSL_Insecure` property to `1`.

8. Optionally, set the `partner_spid` property to a partner SPID (service provider ID) value.

Configuring SSL Verification in a Non-Windows Machine

If you are connecting to a Athena server that has Secure Sockets Layer (SSL) enabled, you can configure the connector to connect to an SSL-enabled socket. When using SSL to connect to a server, the connector can be configured to verify the identity of the server.

You can set the connection properties described below in a connection string or in a DSN (in the `odbc.ini` file). Settings in the connection string take precedence over settings in the DSN.

To configure SSL verification on a non-Windows machine:

1. To enable SSL connections, set the `SSL` attribute to `1`.

2. To allow the common name of a CA-issued SSL certificate to not match the host name of the Athenaserver, set the `Mismatch` attribute to `1`.

3. Choose one:

    *   To configure the connector to load SSL certificates from a specific `.pem` file when verifying the server, set the `TrustedCerts` attribute to the full path of the `.pem` file.

- Or, to use the trusted CA certificates `.pem` file that is installed with the connector, do not specify a value for the `TrustedCerts` attribute.

Configuring Proxy Connections on Non-Windows Machines

You can configure the connector to connect through a proxy server instead of connecting directly to the Athena service.

> ⚠ **Important:**
> If you are connecting to Athena through a proxy server, make sure that the proxy server does not block port 444. The result set streaming API uses port 444 on the Athena server for outbound communications. For more information, see Use Result Set Streaming.

You can set the connection properties described below in a connection string or in a DSN (in the `odbc.ini` file). Settings in the connection string take precedence over settings in the DSN.

To configure a proxy connection on a non-Windows machine:

1. To enable proxy connections, set the `UseProxy` property to `1`.

2. Set the `ProxyHost` property to the IP address or host name of your proxy server.

3. Set the `ProxyPort` property to the number of the TCP port that the proxy server uses to listen for client connections.

4. Set the `ProxyUID` property to your user name for accessing the proxy server.

5. Set the `ProxyPWD` property to your password for accessing the proxy server.

If the proxy server is configured to intercept SSL-encrypted connections, then in addition to specifying the proxy server information described above, you must also export the proxy server's root certificate onto your machine and configure the connector to use it.

To export and specify the proxy server's root certificate:

1. Export the proxy's certificate as a `.pem` file. You can do this using OpenSSL.

   If necessary, you can export the certificate as another format, such as `.cer`, and convert that file into a `.pem` file.

   For example:

   openssl x509 -inform der -in certificate.cer -out certificate.pem

2. In your connection string or DSN (in the `odbc.ini` file), set the `TrustedCerts` property to the full path and name of `.pem` file containing the proxy server's root certificate.

Configuring Query Result Encryption on a Non-Windows Machine

You can configure the Simba Amazon Athena ODBC Connector to encrypt your query results using any of the encryption protocols that Athena supports.

You can set the connection properties described below in a connection string or in a DSN (in the `odbc.ini` file). Settings in the connection string take precedence over settings in the DSN.

To configure query result encryption on a non-Windows machine:

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

> **Note:** For detailed information about these encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*: http://docs.aws.amazon.com/athena/latest/ug/encryption.html.

Option Name	Description
NOT_SET	The connector does not encrypt the data.
SSE_S3	The connector uses server-side encryption with an Amazon S3-managed key.
SSE_KMS	The connector uses server-side encryption with an AWS KMS-managed key.
CSE_KMS	The connector uses client-side encryption with an AWS KMS-managed key.

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

Configuring Logging Options in a Non-Windows Machine

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.

You can set the connection properties described below in a connection string, in a DSN (in the `odbc.ini` file), or as a connector-wide setting (in the `simba.athenaodbc.ini` file). Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To enable logging on a non-Windows machine:

1. To specify the level of information to include in log files, set the `LogLevel` property to one of the following numbers:

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

LogLevel Value	Description
6	Logs all connector activity.

> **Note:** If `UseAwsLogger` is set to 1, the connector also logs information from AWS API calls.

2. Set the `LogPath` key to the full path to the folder where you want to save log files.

3. Set the `LogFileCount` key to the maximum number of log files to keep.

> **Note:** After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

4. Set the `LogFileSize` key to the maximum size of each log file in bytes.

> **Note:** After the maximum file size is reached, the connector creates a new file and continues logging.

5. Optionally, to prefix the log file name with the user name and process ID associated with the connection, set the `UseLogPrefix` property to `1`.

6. Optionally, to include information about AWS API calls in the log, set `UseAwsLogger` to `1`.

7. Save the `simba.athenaodbc.ini` configuration file.

8. Restart your ODBC application to make sure that the new settings take effect.

The Simba Amazon Athena ODBC Connector produces the following log files at the location you specify using the `LogPath` key:

- A `simbaathenaodbcdriver.log` file that logs connector activity that is not specific to a connection.

- A `simbaathenaodbcdriver_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 you set the `UseLogPrefix` property to `1`, then each file name is prefixed with *[UserName]_ [ProcessID]_*, where *[UserName]* is the user name associated with the connection and *[ProcessID]* is the process ID of the application through which the connection is made. For more information, see UseLogPrefix.

To disable logging on a non-Windows machine:

1. Set the `LogLevel` key to `0`.

2. Save the `simba.athenaodbc.ini` configuration file.

3. Restart your ODBC application to make sure that the new settings take effect.

Testing the Connection in Non-Windows Machine

To test the connection, you can use an ODBC-enabled client application. For a basic connection test, you can also use the test utilities that are packaged with your driver manager installation. For example, the iODBC driver manager includes simple utilities called `iodbctest` and `iodbctestw`. Similarly, the unixODBC driver manager includes simple utilities called isql and iusql.

Using the iODBC Driver Manager

You can use the iodbctest and iodbctestw utilities to establish a test connection with your connector. Use iodbctest to test how your connector works with an ANSI application, or use iodbctestw to test how your connector works with a Unicode application.

> **Note:** There are 32-bit and 64-bit installations of the iODBC driver manager available. If you have only one or the other installed, then the appropriate version of iodbctest (or iodbctestw) is available. However, if you have both 32- and 64-bit versions installed, then you need to make sure that you are running the version from the correct installation directory.

For more information about using the iODBC driver manager, see http://www.iodbc.org.

To test your connection using the iODBC driver manager:

1. Run **iodbctest** or **iodbctestw**.

2. Optionally, if you do not remember the DSN, then type a question mark (`?`) to see a list of available DSNs.

3. Type the connection string for connecting to your data store, and then press ENTER. For more information, see Using a Connection String.

   If the connection is successful, then the `SQL>` prompt appears.

Using the unixODBC Driver Manager

You can use the isql and iusql utilities to establish a test connection with your connector and your DSN. isql and iusql can only be used to test connections that use a DSN. Use isql to test how your connector works with an ANSI application, or use iusql to test how your connector works with a Unicode application.

> **Note:** There are 32-bit and 64-bit installations of the unixODBC driver manager available. If you have only one or the other installed, then the appropriate version of isql (or iusql) is available. However, if you have both 32- and 64-bit versions installed, then you need to make sure that you are running the version from the correct installation directory.

For more information about using the unixODBC driver manager, see http://www.unixodbc.org.

To test your connection using the unixODBC driver manager:

- Run isql or iusql by using the corresponding syntax:

  ```
  ■    isql [DataSourceName]

  ■    iusql [DataSourceName]
  ```

[DataSourceName] is the DSN that you are using for the connection.

If the connection is successful, then the `SQL>` prompt appears.

> **Note:** For information about the available options, run isql or iusql without providing a DSN.

Using a Connection String

For some applications, you might need to use a connection string to connect to your data source. For detailed information about how to use a connection string in an ODBC application, refer to the documentation for the application that you are using.

The connection strings in the following sections are examples showing the minimum set of connection attributes that you must specify to successfully connect to the data source. Depending on the configuration of the data source and the type of connection you are working with, you might need to specify additional connection attributes. For detailed information about all the attributes that you can use in the connection string, see Connector Configuration Properties.

DSN Connection String Example

The following is an example of a connection string for a connection that uses a DSN:

DSN=*[DataSourceName]*

[DataSourceName] is the DSN that you are using for the connection.

You can set additional configuration options by appending key-value pairs to the connection string. Configuration options that are passed in using a connection string take precedence over configuration options that are set in the DSN.

DSN-less Connection String Examples

Some applications provide support for connecting to a data source using a connector without a DSN. To connect to a data source without using a DSN, use a connection string instead.

The placeholders in the examples are defined as follows, in alphabetical order:

- *[CertificateStorePath]* is the complete path to the proxy's certificate.

- *[ClientID]* is the the Client ID to use when authenticating the connection using the Azure AD service.

- *[ClientSecret]* is the the Client Secret to use when authenticating the connection using the Azure AD service.

- *[CredProviderHost]* is the host name of your credentials provider service.

- *[ProxyHost]* is the IP address of the proxy server you are connecting through.

- *[Proxy Password]* is the password associated with the application's proxy user ID.

- *[ProxyUserID]* is the ID your application uses to log into the proxy server.

- *[Region]* is the AWS region of the Athena instance that you want to connect to.

- *[RoleSessionName]* is the name of the assumed role session.

- *[SAMLURL]* is the URL for the resource on the identity provider's website when using the SAML services through a browser plugin.

- *[S3Path]* is the path of the Amazon S3 location where you want to store query results, prefixed by `s3://`.

- *[VPC Endpoint]* is the endpoint URL for your Virtual Private Cloud.

- *[YourAccessKey]* is the access key provided by your AWS account.

- *[YourCredProviderPassword]* is your password for your credentials provider service.

- *[YourCredProviderUserName]* is your user name for your crendetials provider service.

- *[YourProfileName]* is the name of the IAM profile to use for authentication.

- *[YourSecretKey]* is the secret key provided by your AWS account.

- *[YourTenantID]* is the the Azure AD-provided unique ID associated with your Athena application.

- *[WebIdentityToken]* is the JSON web token generated after OAuth authentication from Azure Active directory.

Connecting to Athena Using the Default Credentials Provider Chain

The following is the format of a DSN-less connection string for connecting to Athena using the default credentials provider chain:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation=*[S3Path]*;AuthenticationType=Default Credentials;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=Default Credentials;

Connecting to Athena Using a VPC Endpoint

The following is the format of a DSN-less connection string for connecting to Athena using ta VPC Endpoint:

Driver=Simba Amazon Athena ODBC Connector;EndpointOverride=*[VPC Endpoint]*;AwsRegion=*[Region]*;S3OutputLocation=*[S3Path]*;AuthenticationType=IAM Profile;AWSProfile=*[YourProfileName]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;EndpointOverride=vpce-123456abc.athena.us-east-1.vpce.amazonaws.com:443;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=AuthenticationType=IAM Profile;AWSProfile=simba;

Connecting to Athena Using IAM Credentials

The following is the format of a DSN-less connection string for connecting to Athena using IAM credentials:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation= *[S3Path]*;AuthenticationType=IAM Credentials;UID=*[YourAccessKey]*;PWD=*[YourSecretKey]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=IAM Credentials;UID=ABCABCABC123ABCABC45;PWD=abCD+E1f2Gxhi3J4kImN/OP5QrSTuvwXYzabcdEF;

Connecting to Athena Using an IAM Profile

The following is the format of a DSN-less connection string for connecting to Athena using an IAM profile:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation= *[S3Path]*;AuthenticationType=IAM Profile;AWSProfile=*[YourProfileName]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=IAM Profile;AWSProfile=simba;

Connecting to Athena Using an Instance Profile

The following is the format of a DSN-less connection string for connecting to Athena using an instance profile from the Amazon EC2 Instance Metadata Service:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation= *[S3Path]*;AuthenticationType=Instance Profile;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=Instance Profile;

Connecting to Athena Using the AD FS Credentials Provider

The following is the format of a DSN-less connection string for connecting to Athena using credentials provided by the AD FS service. If you are connecting to Athena from a Windows machine, the `UID` and `PWD` properties are optional.

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation= *[S3Path]*;AuthenticationType=ADFS;IdP_Host=*[CredProviderHost]*;UID= *[YourCredProviderUserName]*;PWD=*[YourCredProviderPassword]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=ADFS;IdP_Host=example.adfs.server;UID=HOME\jsmith;PWD=simba12345;

Connecting to Athena Using the Azure AD Credentials Provider

The following is the format of a DSN-less connection string for connecting to Athena using credentials provided by the Azure AD service.

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation= *[S3Path]*;AuthenticationType=AzureAD;UID=*[YourCredProviderUserName]*;PWD= *[YourCredProviderPassword]*;idp_tenant=*[YourTenantID]*;client_id=*[ClientID]*;client_secret= *[ClientSecret]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=AzureAD;UID=jsmith;PWD=simba12345;idp_tenant=xyz;client_id=xyz;client_secret=xyz;

Connecting to Athena Using the Browser Azure AD Credentials Provider

The following is the format of a DSN-less connection string for connecting to Athena using credentials provided by the Browser Azure AD service.

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation= *[S3Path]*;AuthenticationType=BrowserAzureAD;UID=*[YourCredProviderUserName]*;PWD= *[YourCredProviderPassword]*;idp_tenant=*[YourTenantID]*;client_id=*[ClientID]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=BrowserAzureAD;UID=jsmith;PWD=simba12345;idp_tenant=xyz;client_id=xyz;

Connecting to Athena Using the Browser SAML Credentials Provider

The following is the format of a DSN-less connection string for connecting to Athena using credentials provided by the Browser SAML service.

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation= *[S3Path]*;AuthenticationType=BrowserSAML;UID=*[YourCredProviderUserName]*;PWD= *[YourCredProviderPassword]*;Login_URL=*[SAMLURL]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=BrowserSAML;UID=jsmith;PWD=simba12345;Login_URL=http://localhost:abc/athena;

Connecting to Athena Using a JSON Web Token (JWT)

The following is the format of a DSN-less connection string for connecting to Athena using a JWT.

Driver=Simba Amazon Athena ODBC Connector;AuthenticationType=JWT;web_identity_token=*[WebIdentityToken]*;preferred_role=*[IAMRole]*;role_session_name=*[RoleSessionName]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AuthenticationType=JWT;web_identity_token=bjsdbhcjsdchsdchsdc;Preferred_Role=arn:aws:iam::187862086336:role/ADFS-Dev;role_session_name=athena

Connecting to Athena Using the Okta Credentials Provider

The following is the format of a DSN-less connection string for connecting to Athena using credentials provided by the Okta credentials service.

Driver=Simba Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation=*[S3Path]*;AuthenticationType=Okta;IdP_Host=*[CredProviderHost]*;UID=*[YourCredProviderUserName]*;PWD=*[YourCredProviderPassword]*;App_ID=*[YourOktaAppId]*;

For example:

Driver=Simba Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=Okta;IdP_Host=dev-123456.okta.com;UID=jsmith@example.com;PWD=simba12345;App_ID=12abc123456789/123;

Connecting to Athena Using the PingFederate Service

The following is the format of a DSN-less connection string for connecting to a Athena server using the PingFederate service:

Driver=Simba Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation=*[S3Path]*;AuthenticationType=Ping;IdP_Host=*[CredProviderHost]*;UID=*[YourCredProviderUserName]*;PWD=*[YourCredProviderPassword]*;Preferred_Role=*[IAMRole]*;

For example:

Driver=Simba Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=Ping;IdP_Host=dev-123456.okta.com;UID=jsmith@example.com;PWD=simba12345;IdP_Host=ping.simba.com;Preferred_Role=dbAdmin;

Connecting to Athena Using a Proxy Server

The following is the format of a DSN-less connection string for connecting to Athena using a proxy server:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation=*[S3Path]*;AuthenticationType=Default Credentials;UseProxy=1;ProxyScheme=HTTPS;ProxyHost=*[ProxyHost]*;ProxyPort=*[Port]*;ProxyUID=*[ProxyUserID]*;ProxyPWD=*[ProxyPassword]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=Default Credentials;UseProxy=1;ProxyScheme=HTTPS;ProxyHost=123.456.789.012;ProxyPort=8080;ProxyUID=simba;ProxyPWD=simba;

Connecting to Athena Using a Proxy Server on a Non-Windows Machine With Trusted Certificate

The following is the format of a DSN-less connection string for connecting to Athena using a proxy server:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=*[Region]*;S3OutputLocation= *[S3Path]*;AuthenticationType=Default Credentials;UseProxy=1;ProxyScheme=HTTPS;ProxyHost= [Proxy Host];ProxyPort=[Port];ProxyUID=[Proxy User ID];ProxyPWD=*[Proxy Password]*;TrustedCerts= *[CertificateStorePath]*;

For example:

Driver=Simba Amazon Athena ODBC Connector;AwsRegion=us-east-2;S3OutputLocation=s3://query-results-bucket/test-folder-1;AuthenticationType=Default Credentials;UseProxy=1;ProxyScheme=HTTPS;ProxyHost=123.456.789.012;ProxyPort=8080;ProxyUID=simba;ProxyPWD=simba;TrustedCerts=/disk/dir/certificates.pem;

Example: Using Workgroups

A workgroup is an Athena feature that enables you to control the data access and costs associated with running queries. For more information, see "Using Workgroups to Control Query Access and Costs" in the Amazon Athena User Guide: https://docs.aws.amazon.com/athena/latest/ug/manage-queries-control-costs-withworkgroups.html.

To use a workgroup when connecting to Athena through the Simba Amazon Athena ODBC Connector, either specify a workgroup in the DSN, write a connection URL that sets the `Workgroup` property to the name of your workgroup. For example, to use a workgroup named `SimbaAdmins`:

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

Features

For more information on the features of the Simba Amazon Athena ODBC Connector, see the following:

- Catalog and Schema Support

- File Formats

- Data Types

- Result Set Streaming Support

- Query Execution Polling

- Security and Authentication

Catalog and Schema Support

The Simba Amazon Athena ODBC Connector supports both catalogs and schemas to make it easy for the connector to work with various ODBC applications.

Amazon Athena provides catalogs that enable you to access the data source that is being queried. These catalogs contain databases, which correspond to ODBC schemas, and each database contains data that has been organized into tables.

By default, if your query statement does not specify a catalog, the connector queries the data under the catalog named AwsDataCatalog. You can use the Catalog connection property to specify a different default catalog for your queries. For more information, see Catalog.

File Formats

The Simba Amazon Athena ODBC Connector supports all the file formats that Athena supports, which include the following:

- Avro

- Comma-Separated Values (CSV)

- JavaScript Object Notation (JSON)

- Optimized Row Columnar (ORC)

- Parquet

Data Types

The Simba Amazon Athena ODBC Connector supports many common data formats, converting between Athena data types and SQL data types.

The following table lists the supported data type mappings.

Athena Type	SQL Type
ARRAY	- SQL_VARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is disabled. - SQL_WVARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is enabled.
BIGINT	SQL_BIGINT
BINARY	SQL_VARBINARY
BOOLEAN	SQL_BIT
CHAR	- SQL_CHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is disabled. - SQL_WCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is enabled.
DATE ℹ **Note:** Not supported for Parquet files.	- SQL_TYPE_DATE if the application uses ODBC version 3.00 or later. - SQL_DATE if the application uses an ODBC version earlier than 3.00.
DECIMAL (p, s)	SQL_DECIMAL
DOUBLE	SQL_DOUBLE
FLOAT	SQL_REAL
INTEGER ℹ **Note:** Although Athena reports integer data as type INT, the connector reports integer data as type INTEGER to ensure compatibility with standard BI tools. For more information, see Integer Support.	SQL_INTEGER
MAP	SQL_VARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is disabled.

Athena Type	SQL Type
	■ SQL_WVARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is enabled.
SMALLINT	SQL_SMALLINT
STRING	■ SQL_VARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is disabled. ■ SQL_WVARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is enabled.
STRUCT	■ SQL_VARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is disabled. ■ SQL_WVARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is enabled.
TIMESTAMP	■ SQL_TYPE_TIMESTAMP if the application uses ODBC version 3.00 or later. ■ SQL_TIMESTAMP if the application uses an ODBC version earlier than 3.00.
TINYINT	SQL_TINYINT
VARCHAR	■ SQL_VARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is disabled. ■ SQL_WVARCHAR if the Use SQL Unicode Types option (the `UseSQLUnicodeTypes` property) is enabled.

Integer Support

Athena combines two different implementations of the integer data type:

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

- In all other queries, Athena uses the INTEGER data type from Presto.

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

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

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

Result Set Streaming Support

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

This is configured using the Use Result Set Streaming option (the `UseResultSetStreaming` property). For more information, see Use Result Set Streaming.

Query Execution Polling

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

- Max Query Execution Polling Interval

- Min Query Execution Polling Interval

- QueryExecutionPollingIntervalMultiplier

The connector starts by polling the server every minimum interval, which is the number of milliseconds specified in Min Query Execution Polling Interval. The connector then increases the polling interval using the multiplier specified in Query Execution Polling Interval Multiplier until the interval specified in Max Query Execution Polling Interval is reached. The connector then continues to poll the server using this maximum interval until the query results are returned.

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

- Min Query Execution Polling Interval: 5

- Max Query Execution Polling Interval: 1800000

- Query Execution Polling Interval Multiplier: 2

Using these settings, the connector polls the server 5ms after the query begins to run, and then doubles the interval between polls after each subsequent poll. In other words, the connector polls the server at these intervals: 5ms after the query starts to run, 10ms after the first poll, 20ms after the second poll, and so on until the 1800000ms polling interval is reached. The connector then continues to poll the server every 1800000ms until the query results are returned.

Security and Authentication

To protect data from unauthorized access, Athena requires all connections to be authenticated using IAM credentials (an access key and a secret key), and uses the SSL protocol that is implemented in Amazon Web Services. The Simba Amazon Athena ODBC Connector protects your data by providing support for these authentication protocols and further obscuring data from unwanted access by providing encryption options for your query results.

The connector can authenticate your connection using IAM credentials from any of the following sources:

- A default credentials provider chain

- An IAM profile

- An instance profile

- The DSN or connection string settings

- The Active Directory Federation Services (AD FS) credentials provider

- Okta service credentials provider

For detailed configuration instructions, see Configuring Authentication in Windows or Configuring Authentication on Non-Windows Machines.

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

For query results, the Simba Amazon Athena ODBC Connector supports all the encryption options that Athena supports. For detailed information about the supported encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*: http://docs.aws.amazon.com/athena/latest/ug/encryption.html. For information about configuring encryption in the connector, see Creating a Data Source Name in Windows or Configuring Query Result Encryption on a Non-Windows Machine.

Connector Configuration Properties

Connector Configuration Options lists the configuration options available in the Simba Amazon Athena ODBC Connector alphabetically by field or button label. Options having only key names, that is, not appearing in the user interface of the connector, are listed alphabetically by key name.

When creating or configuring a connection from a Windows machine, the fields and buttons described below are available in the following dialog boxes:

- Simba Athena ODBC Driver DSN Setup

- Authentication Options

- Advanced Options

- Logging Options

When using a connection string or configuring a connection from a non-Windows machine, use the key names provided below.

Configuration Options Appearing in the User Interface

The following configuration options are accessible via the Windows user interface for the Simba Amazon Athena ODBC Connector, or via the key name when using a connection string or configuring a connection from a Linux or macOS computer:

- Allow Common Name Host Name Mismatch

- AuthenticationType

- AwsProfile

- AwsRegion

- Binary Column Length

- Catalog

- Client ID

- Client Secret

- Duration

- EnableResultReuse

- Okta MFA Wait Time

- Okta Phone Number

- Using PingFederate Service

- Partner SPID

- Password

- Preferred_Role

- Proxy Host

- Proxy Password

- Proxy Port

- Proxy Username

- QueryExecutionPollingIntervalMultiplier

- S3OutputEncOption
- Endpoint Override
- GlueEndpointOverride
- IdP_Host
- IdP_Port
- KMS Key
- LakeformationEnabled
- Listen_Port
- Log Level
- Log Path
- Login_URL
- MaxCatalogNameLen
- Max Column Name Length
- Max Complex Type Column Length
- Max File Size
- Max Number Files
- Max Query Execution Polling Interval
- Max Schema Name Length
- Max Table Name Length
- Metadata Retrieval Method
- Min Query Execution Polling Interval
- Non Proxy Hosts
- Okta App ID
- Okta App Name

- Result Reuse Max Time
- Role Session Name
- Rows To Fetch Per Block
- S3 Output Location
- Schema
- Session Token
- SSL Insecure
- Streaming Endpoint Override
- String Column Length
- STS Endpoint Override
- Tenant_ID
- Timeout
- Trusted CA Certificate
- UseAwsLogger
- Use HTTP Proxy For IdP Host
- Use Proxy
- Use Result Set Streaming
- Use SQL Unicode Types
- Use System Trust Store
- User
- Verify SSL
- Web Token
- Workgroup

- **Okta MFA Type**

Allow Common Name Host Name Mismatch

This option specifies whether a CA-issued SSL certificate name must match the host name of the Athena server.

> **ⓘ Note:** The key for this option used to be `CAIssuedCertNamesMismatch`, and is still recognized by the connector under that key. If both keys are defined, `AllowHostNameCNMismatch` will take precedence.
>
> This setting is applicable only when SSL is enabled.

- Enabled (`1`): The connector allows a CA-issued SSL certificate name to not match the host name of the Athena server.

- Disabled (`0`): The CA-issued SSL certificate name must match the host name of the Athena server.

Key Name	Default Value	Required
`AllowHostNameCNMismatch`	**Clear** (`0`)	No

AuthenticationType

This option specifies how the connector authenticates the connection to Athena.

- **Default Credentials** (`Default Credentials`): The connector authenticates the connection using credentials that are stored in one of the locations in the default credentials provider chain.

- **IAM Credentials** (`IAM Credentials`): The connector authenticates the connection using an access key and a secret key that is specified directly in the connection information.

- **IAM Profile** (`IAM Profile`): The connector authenticates the connection using credentials that are associated with an IAM profile in a credentials file.

- **Instance Profile** (`Instance Profile`): The connector authenticates the connection using credentials that have been loaded from the Amazon EC2 Instance Metadata Service into an instance profile.

- **ADFS** (`ADFS`): The connector authenticates the connection using credentials provided by the Active Directory Federation Services (AD FS) credential provider.

- **JWT** (`JWT`): The connector authenticates the connection using a JSON Web Token (JWT).

- **Okta** (`Okta`): The connector authenticates the connection using credentials provided by the Okta credential provider.

- Ping (`Ping`): The connector authenticates the connection using credentials stored in the PingFederate service.

Key Name	Default Value	Required
AuthenticationType	IAM Credentials (`IAM Credentials`)	Yes

AwsProfile

The name of the profile to use from the credentials file. This setting is applicable only when Authentication Type is set to IAM Profile (the `AuthenticationType` property is set to `IAM Profile`).

Key Name	Default Value	Required
AwsProfile	default	No

AwsRegion

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

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

Key Name	Default Value	Required
AwsRegion	None	Yes

Binary Column Length

The maximum data length for BINARY columns.

Key Name	Default Value	Required
BinaryColumnLength	32767	No

Catalog

The default catalog used for query execution. If no catalog is specified in the SQL query, the Athena connector executes the query against the catalog specified.

Key Name	Default Value	Required
Catalog	AwsDataCatalog	No

Client ID

The Client ID to use when authenticating the connection using the Azure AD services.

Key Name	Default Value	Required
client_id	None	Yes, if authenticating through the Azure AD service.

Client Secret

The Client Secret to use when authenticating the connection using the Azure AD service.

Key Name	Default Value	Required
client_secret	None	No

Duration

The duration, in seconds, of the role session.

Key Name	Default Value	Required
duration	0	No

EnableResultReuse

This option specifies whether the connector reuses the results of previously run queries.

- `true`: The connector reuses the results of previously run queries.

- `false`: The connector does not reuse the results of previously run queries.

Key Name	Default Value	Required
EnableResultReuse	false	No

S3OutputEncOption

The encryption protocol that the connector uses to encrypt your query results.

- NOT_SET (`NOT_SET`): The connector does not encrypt the data.

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

- SSE_KMS (`SSE_KMS`): The connector uses server-side encryption with an AWS KMS-managed key.

- CSE_KMS (`CSE_KMS`): The connector uses client-side encryption with an AWS KMS-managed key.

For detailed information about these encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*: http://docs.aws.amazon.com/athena/latest/ug/encryption.html.

Key Name	Default Value	Required
S3OutputEncOption	NOT_SET (NOT_SET)	Yes

Endpoint Override

The endpoint for the Athena instance the connector connects to if not using the default endpoint. If this property is not set, the connector attempts to connect to the default Athena endpoint.

Key Name	Default Value	Required
EndpointOverride	None	Yes, if not using the default Athena endpoint.

GlueEndpointOverride

The endpoint to be used for make AWS glue API calls. This property should not contain the prefix `http://` or `https://`.

Key Name	Default Value	Required
GlueEndpointOverride	None	Yes, if you are not using the default Athena endpoint, and want to retrieve metadata from the Glue service.

IdP_Host

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

Key Name	Default Value	Required
IdP_Host	None	Yes, if authenticating through AD FS, Okta, or Ping.

IdP_Port

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

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

Key Name	Default Value	Required
IdP_Port	443	No

KMS Key

The KMS customer key to use when encrypting query results using SSE_KMS or CSE_
KMS encryption.

For detailed information about the supported encryption options, see "Configuring Encryption Options" in the *Amazon Athena User Guide*: http://docs.aws.amazon.com/athena/latest/ug/encryption.html.

Key Name	Default Value	Required
S3OutputEncKMSKey	None	Yes, if using SSE_KMS or CSE_ KMS encryption.

LakeformationEnabled

This option specifies whether the connector retrieves SAML credentials using Lake Formation service.

- `true`: The connector retrieves SAML credentials using Lake Formation service.

- `false`: The connector retrieves SAML credentials using STS service.

Key Name	Default Value	Required
LakeformationEnabled	false	No

Lake formation Endpoint Override

The endpoint for the Lake Formation service when using the assumeDecoratedRoleWithSaml API to retrieve temporary credentials.

Key Name	Default Value	Required
LakeformationEndpointOverride	None	No

Listen_Port

The port that the connector uses to receive the SAML response from the identity provider when using the SAML services through a browser plugin.

Key Name	Default Value	Required
Listen_Port	7890	No

Log Level

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.
> - When logging with connection strings and DSNs, this option only applies to per-connection logs.

Set the property to one of the following values:

- OFF (0): Disable all logging.

- FATAL (1): Logs severe error events that lead the connector to abort.

- ERROR (2): Logs error events that might allow the connector to continue running.

- WARNING (3): Logs events that might result in an error if action is not taken.

- INFO (4): Logs general information that describes the progress of the connector.

- DEBUG (5): Logs detailed information that is useful for debugging the connector.

- TRACE (6): Logs all connector activity.

When logging is enabled, the connector produces the following log files at the location you specify in the Log Path (`LogPath`) property:

- A `simbaathenaodbcdriver.log` file that logs connector activity that is not specific to a connection.

- A `simbaathenaodbcdriver_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 you enable the `UseLogPrefix` connection property, the connector prefixes the log file name with the user name associated with the connection and the process ID of the application through which the connection is made. For more information, see UseLogPrefix.

Key Name	Default Value	Required
LogLevel	OFF (0)	No

Log Path

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

> **Important:** When logging with connection strings and DSNs, this option only applies to per-connection logs.

Key Name	Default Value	Required
LogPath	None	Yes, if logging is enabled.

Login_URL

The URL for the resource on the identity provider's website when using the SAML services through a browser plugin.

Key Name	Default Value	Required
Login_URL	None	Yes, if authenticating with the SAML services through a browser plugin.

Login To RP

The endpoint for the ADFS auth plugin to use custom LoginToRP parameter.

Key Name	Default Value	Required
LoginToRP	None	No

MaxCatalogNameLen

This option specifies the maximum number of characters that the driver reports for SQL_MAX_ CATALOG_NAME_LEN. This allows the applications to allocate a large enough buffer to retrieve the catalog name.

This option can be set to any integer from 0 to 65535, inclusive. To indicate that there is no maximum length or that the length is unknown, set this option to 0.

Key Name	Default Value	Required
MaxCatalogNameLen	0	No

Max Column Name Length

This option specifies the maximum number of characters that the driver reports for SQL_MAX_ COLUMN_NAME_LEN. This allows the applications to allocate a large enough buffer to retrieve the column name.

This option can be set to any integer from 0 to 65535, inclusive. To indicate that there is no maximum length or that the length is unknown, set this option to 0.

Key Name	Default Value	Required
MaxColumnNameLen	0	No

Max Complex Type Column Length

The maximum data length for complex data types that the connector casts to SQL_VARCHAR. For example, ARRAY, MAP, and STRUCT data types.

Key Name	Default Value	Required
ComplexTypeColumnLength	65535	No

Max File Size

The maximum size of each log file in bytes. After the maximum file size is reached, the connector creates a new file and continues logging.

If this property is set using the Windows UI, the entered value is converted from megabytes (MB) to bytes before being set.

Important: When logging with connection strings and DSNs, this option only applies to per-connection logs.

Key Name	Default Value	Required
LogFileSize	20971520	No

Max Number Files

The maximum number of log files to keep. After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

Important: When logging with connection strings and DSNs, this option only applies to per-connection logs.

Key Name	Default Value	Required
LogFileCount	50	No

Max Query Execution Polling Interval

The maximum time, in milliseconds, to wait between attempts when polling the server for the query execution result. This value cannot be lower than the Min Query Execution Polling Interval setting.

For more information Query Execution Polling.

Key Name	Default Value	Required
MaxQueryExecutionPollingInterval	1800000	No

Max Schema Name Length

This option specifies the maximum number of characters that the driver reports for SQL_MAX_ SCHEMA_NAME_LEN. This allows the applications to allocate a large enough buffer to retrieve the schema name.

This option can be set to any integer from 0 to 65535, inclusive. To indicate that there is no maximum length or that the length is unknown, set this option to `0`.

Key Name	Default Value	Required
MaxSchemaNameLen	256	No

Max Table Name Length

This option specifies the maximum number of characters that the driver reports for SQL_MAX_TABLE_NAME_LEN. This allows the applications to allocate a large enough buffer to retrieve the table name.

This option can be set to any integer from 0 to 65535, inclusive. To indicate that there is no maximum length or that the length is unknown, set this option to `0`.

Key Name	Default Value	Required
MaxTableNameLen	0	No

Metadata Retrieval Method

This property determines how the metadata is retrieved from Athena for different ODBC API calls, such as getTables or getColumns.

- `Auto`: At connection time connector automatically determines whether to use AWS Glue or Query to retrieve metadata for the specified Athena region. If AWS Glue is supported in the region and Athena has been upgraded to use AWS Glue, the connector uses AWS Glue to retrieve the metadata. If AWS Glue is not supported in the region or Athena hasn't been upgraded to use AWS Glue, the connector queries Athena to retrieve the metadata.

- `Glue`: The connector uses AWS Glue to retrieve the metadata regardless of whether AWS Glue is supported or used in the region.

- `ProxyAPI`: The connector uses Athena's proxy API. This is used to query external catalogs.

- `Query`: The connector uses Query to retrieve the metadata regardless of whether AWS Glue is supported or used in that region.

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

Key Name	Default Value	Required
MetadataRetrievalMethod	Auto	No

Min Query Execution Polling Interval

The minimum value of the polling interval, in milliseconds. This value must be greater than 0 but less than Max Query Execution Polling Interval. A value of `0` or a negative value means that the default value of `5` is used.

For more information see Query Execution Polling.

Key Name	Default Value	Required
MinQueryExecutionPollingInterval	5	No

Non Proxy Hosts

A list of hosts separated by a comma (,), that the connector can access without connecting through the proxy server, when a proxy connection is enabled.

> **Note:** When specifying patterns, use dots (.) instead of asterisks (*).

Key Name	Default Value	Required
NonProxyHosts	None	No

Okta App ID

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

Key Name	Default Value	Required
App_ID	None	Yes, if authenticating through the Okta service.

Okta App Name

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

Key Name	Default Value	Required
App_Name	None	No

Okta MFA Type

The factor type when using Okta MFA authentication, from the following list:

- **oktaverifywithtotp:**

  If the user has already enrolled with Okta MFA TOTP factor, the user must enter the password in the dialog box before querying the data store.

  If the user has not previously enrolled with Okta MFA TOTP factor, the user must use the Okta Verify app to scan the QR code displayed on the browser. Then, the user must enter the password from the app in the dialog box to activate their device.

- **oktaverifywithpush:**

  If the user has already enrolled with Okta MFA push factor, a push notification is sent to the user's device. Once the push notification is approved, the user can query the data store.

  If the user has not previously enrolled with Okta MFA push factor, the user must use the Okta Verify app to scan the QR code displayed on the browser.

- **SmsAuthentication:**

  If the user has already enrolled with SMS Authentication, the user must enter the password sent to their mobile number in the dialog box before querying the data store.

  If the user has not previously enrolled with SMS Authentication, the user must enter the password sent to their mobile number in the dialog box to register the device.

- **GoogleAuthenticator:**

  If the user has already enrolled with Okta MFA Google Authenticator, the user must enter the password in the dialog box before querying the data store.

  If the user has not previously enrolled with Okta MFA Google Authenticator, the user must use the Google Authenticator app to scan the QR code displayed on the browser. Then, the user must enter the password from the app in the dialog box to activate their device.

Key Name	Default Value	Required
okta_mfa_type	None	Yes, if MFA is required for Okta Authentication.

Okta MFA Wait Time

The MFA timeout value, in seconds.

> **Note:**
> If the value is less than 20 seconds, the connector returns an error.

Key Name	Default Value	Required
okta_mfa_wait_time	60	No

Okta Phone Number

The phone number used to receive a one-time password for SMS Authentication.

> **(i) Note:**
> Okta only supports a US or Canadian number. If a number is not preceded with the country code, +1 is automatically added.

Key Name	Default Value	Required
okta_phone_number	None	Yes, if the Okta MFA type is SMS Authentication.

Partner SPID

The partner SPID (service provider ID) value to use when authenticating the connection using the PingFederate service.

Key Name	Default Value	Required
partner_spid	None	No

Password

If Authentication Type is set to IAM Credentials (the `AuthenticationType` property is set to `IAM Credentials`), then set this property to the secret key provided by your AWS account.

If Authentication Type is set to ADFS or Okta (the `AuthenticationType` property is set to `ADFS` or `Okta`), then set this property to the password that you use to access the credentials service server.

If Authentication Type is set to Ping (the `AuthenticationType` property is set to `Ping`), then set this property to the password that you use to access the PingFederate server.

in Windows machines, if you do not provide a password, the connector attempts to authenticate to the AD FS server using your Windows password over the NTLM protocol.

Key Name	Default Value	Required
PWD	None	Yes, if Authentication Type is set to IAM Credentials, or if the Authentication Type is set to Okta, ADFS, or Ping when connecting from a non-Windows machine.

Preferred_Role

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

Key Name	Default Value	Required
Preferred_Role	None. However, by default, the connector assumes the first role from the list returned in the SAML response from the identity provider.	No

Proxy Host

The host name or IP address of a proxy server that you want to connect through.

Key Name	Default Value	Required
ProxyHost	None	Yes, if connecting through a proxy server.

Proxy Password

The password that you use to access the proxy server.

Key Name	Default Value	Required
ProxyPWD	None	Yes, if connecting to a proxy server that requires authentication.

Proxy Port

The number of the port that the proxy server uses to listen for client connections.

Key Name	Default Value	Required
ProxyPort	8080	No

Proxy Username

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

Key Name	Default Value	Required
ProxyUID	None	Yes, if connecting to a proxy server that requires authentication.

QueryExecutionPollingIntervalMultiplier

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

For more information see Query Execution Polling.

Key Name	Default Value	Required
`QueryExecutionPollingIntervalMultiplier`	2	No

Result Reuse Max Time

The maximum age of previous query results that is considered for reuse.

Key Name	Default Value	Required
`ReusedResultMaxAgeInMinutes`	60	No

Role Session Name

The name of the assumed role session.

Key Name	Default Value	Required
`role_session_name`	`jwt_athena_session`	No

Rows To Fetch Per Block

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

Or, the maximum number of rows to fetch per page if using pagination.

See Use Result Set Streaming for details on result set streaming.

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

Key Name	Default Value	Required
`RowsToFetchPerBlock`	`10000` for result set streaming, `1000` for pagination	No

S3 Output Location

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

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

For details on managing Athena S3 output in workgroups, see https://docs.aws.amazon.com/athena/latest/APIReference/API_ResultConfiguration.html#athena-Type-ResultConfiguration-OutputLocation.

Key Name	Default Value	Required
S3OutputLocation	None	Yes, if the Workgroup property specifies a workgroup that is not configured with an output location.

Schema

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

Key Name	Default Value	Required
Schema	default	No

Session Token

Key Name	Default Value	Required
SessionToken	None	Yes, if you are using temporary security credentials.

SSL Insecure

This property indicates whether the Idp host server certificate should be verified.

- Enabled (`true`): The connector does not check the authenticity of the server certificate.

- Disabled (`false`): The connector checks the authenticity of the server certificate.

Key Name	Default Value	Required
SSL_Insecure	Disabled (`false`)	No

Streaming Endpoint Override

Key Name	Default Value	Required
StreamingEndpointOverride	None	No

STS Endpoint Override

The endpoint to be used for the STS service when using the assumeRoleWithSaml API to retrieve temporary credentials.

Key Name	Default Value	Required
STSEndpointOverride	None	No

String Column Length

The maximum data length for STRING columns.

Key Name	Default Value	Required
StringColumnLength	255	No

Tenant_ID

The Tenant Id associated with your Athena application.

The Azure AD-provided unique ID associated with your Athena application.

Key Name	Default Value	Required
Tenant_Id	None	Yes, if authenticating through the Azure AD service.

Timeout

The maximum amount of time, in seconds, that the connector waits for the redirect URI to fetch the authorization code during Browser Azure AD authentication.

Key Name	Default Value	Required
timeout	120	No

Trusted CA Certificate

The full path and name of the `.pem` file containing the root certificate of the proxy server.

Key Name	Default Value	Required
TrustedCerts	The `cacerts.pem` file in the `/lib` subfolder within the connector's installation directory.	No

UseAwsLogger

This option specifies whether the connector records the log output from any AWS API calls.

- `1`: If logging is enabled, the connector records the log outputs from any AWS API calls in the connector log file.

- `0`: The connector does not log AWS API calls.

> **Note:**
> For information about logging, see Configuring Logging Options in Windows or Configuring Logging Options in a Non-Windows Machine

Key Name	Default Value	Required
`UseAwsLogger`	0	No

Use HTTP Proxy For IdP Host

This option specifies whether the connector accesses the AD FS service through an HTTP proxy.

- Enabled (`1`): The connector accesses the AD FS service through a proxy server based on the information provided in the Proxy Host, Proxy Port, Proxy Username, and Proxy Password fields or the `ProxyHost`, `ProxyPort`, `ProxyUID`, and `ProxyPWD` keys. In order for these proxy settings to take effect, the Use Proxy option (or the `UseProxy` property) must also be enabled.

- Disabled (`0`): The connector accesses the AD FS service directly.

Key Name	Default Value	Required
`UseProxyForIdP`	Disabled (0)	Yes, if authenticating through an AD FS service that must be accessed through an HTTP proxy.

Use Proxy

This option specifies whether the connector uses a proxy server to connect to the data store.

- Enabled (`1`): The connector connects to a proxy server based on the information provided in the Proxy Host, Proxy Port, Proxy Username, and Proxy Password fields or the `ProxyHost`, `ProxyPort`, `ProxyUID`, and `ProxyPWD` keys.

- Disabled (`0`): The connector connects directly to the Athena server.

Key Name	Default Value	Required
`UseProxy`	Clear (0)	No

Use Result Set Streaming

This property specifies whether the connector uses the AWS result set streaming API to fetch result sets.

- `1`: The connector uses the result set streaming API.

- `0`: The connector uses pagination logic for result set fetching.

See Rows To Fetch Per Block to configure how many rows to fetch per stream.

Key Name	Default Value	Required
UseResultsetStreaming	1	No

UseSingleCatalogAndSchema

This option specifies whether the connector returns only single catalog and schema that is specified under connection parameters.

Key Name	Default Value	Required
UseSingleCatalogAndSchema	false	No

Use System Trust Store

This option specifies whether to use a CA certificate from the system trust store, or from a specified `.pem` file.

- Enabled (`1`): The connector verifies the connection using a certificate in the system trust store.

- Disabled (`0`): The connector verifies the connection using a specified `.pem` file. For information about specifying a `.pem` file, see .

  > ℹ **Note:** This option is only available in Windows.

Key Name	Default Value	Required
UseSystemTrustStore		No

Use SQL Unicode Types

This option specifies the SQL types to be returned for string data types.

- Enabled (`1`): The connector returns SQL_WVARCHAR for ARRAY, MAP, STRING, STRUCT, and VARCHAR columns.

- Disabled (`0`): The connector returns SQL_VARCHAR for for ARRAY, MAP, STRING, STRUCT, and VARCHAR columns.

Key Name	Default Value	Required
UseSQLUnicodeTypes	Clear (0)	No

User

If Authentication Type is set to IAM Credentials (the `AuthenticationType` property is set to `IAM Credentials`), then set this property to the access key provided by your AWS account.

If Authentication Type is set to ADFS or Okta (the `AuthenticationType` property is set to `ADFS` or `Okta`), then set this property to the user name that you use to access the authentication server.

When using AD FS, you can include the domain name using the format *[DomainName]\[UserName]*. in Windows machines, if you do not provide a user name when using AD FS, the connector attempts to authenticate to the AD FS server using your Windows user name over the NTLM protocol.

If Authentication Type is set to Ping (the `AuthenticationType` property is set to `Ping`), then set this property to the user name that you use to access the PingFederate service.

Key Name	Default Value	Required
UID	None	Yes, if Authentication Type is set to IAM Credentials, or if the Authentication Type is set to ADFS, Okta, or Ping when connecting from a non-Windows machine.

Verify SSL

This option specifies whether the connector verifies the certificate via the AWS SDK when connecting over SSL.

- Enabled (1): The connector verifies the SSL certificate.

- Disabled (0): The connector does not verify the SSL certificate.

Key Name	Default Value	Required
VerifySSL	Enabled (1)	No

Web Token

The JSON web token generated after OAuth authentication from Azure Active directory.

Key Name	Default Value	Required
web_identity_token	None	Yes, if authenticating using a JSON Web Token (JWT).

Workgroup

The name of the workgroup to use when signing in to Athena.

A workgroup is an Athena feature that enables you to control the data access and costs associated with running queries. For more information, see "Using Workgroups to Control Query Access and Costs" in the Amazon Athena User Guide: https://docs.aws.amazon.com/athena/latest/ug/manage-queries-control-costs-withworkgroups.html.

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

Key Name	Default Value	Required
Workgroup	Primary	No

Configuration Options Having Only Key Names

The following configuration options do not appear in the Windows user interface for the Simba Amazon Athena ODBC Connector. They are accessible only when you use a connection string or configure a connection in macOS or Linux.

- Access_Token

- Driver

- ProxyScheme

The `UseLogPrefix` property must be configured as a Windows Registry key value, or as a connector-wide property in the `simba.athenaodbc.ini` file for macOS or Linux.

- UseLogPrefix

Access_Token

The access token for Browser Azure AD authentication through the Power BI connector.

Key Name	Default Value	Required
Access_Token	None	No

Driver

In Windows, the name of the installed connector for (`Simba Amazon Athena ODBC Connector`).

On other platforms, the name of the installed connector as specified in `odbcinst.ini`, or the absolute path of the connector shared object file.

Key Name	Default Value	Required
`Driver`	`Simba Amazon Athena ODBC Connector` when installed in Windows, or the absolute path of the connector shared object file when installed on a non-Windows machine.	Yes

ProxyScheme

The scheme to use to connect to the proxy server.

Set the property to one of the following values:

- `HTTP`: For connections using HTTP.

- `HTTPS`: For connections using HTTPS.

Key Name	Default Value	Required
`ProxyScheme`	`HTTP`	No

UseLogPrefix

This option specifies whether the connector includes a prefix in the names of log files so that the files can be distinguished by user and application.

Set the property to one of the following values:

- `1`: The connector prefixes log file names with the user name and process ID associated with the connection that is being logged.

  For example, if you are connecting as a user named "jdoe" and using the connector in an application with process ID 7836, the generated log files would be named `jdoe_7836_ simbaathenaodbcdriver.log` and `jdoe_7836_simbaathenaodbcdriver_ connection_[Number].log`, where *[Number]* is a number that identifies each connection-specific log file.

- `0`: The connector does not include the prefix in log file names.

  To configure this option for the Windows connector, you create a value for it in one of the following registry keys:
  - For a 32-bit connector installed on a 64-bit machine: **HKEY_LOCAL_ MACHINE\SOFTWARE\Wow6432Node\Simba\Simba Amazon Athena ODBC Connector\Driver**
  - Otherwise: **HKEY_LOCAL_MACHINE\SOFTWARE\Simba\Simba Amazon Athena ODBC Connector\Driver**

Use `UseLogPrefix` as the value name, and either `0` or `1` as the value data.

To configure this option for a non-Windows connector, you must use the `simba.athenaodbc.ini` file.

Key Name	Default Value	Required
UseLogPrefix	0	No

Third-Party Trademarks

Linux is the registered trademark of Linus Torvalds in Canada, United States and/or other countries.

Mac, macOS, Mac OS, and OS X are trademarks or registered trademarks of Apple, Inc. or its subsidiaries in Canada, United States and/or other countries.

Microsoft, MSDN, Windows, Azure, Windows Server, Windows Vista, and the Windows start button are trademarks or registered trademarks of Microsoft Corporation or its subsidiaries in Canada, United States and/or other countries.

Red Hat, Red Hat Enterprise Linux, and CentOS are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in Canada, United States and/or other countries.

SUSE is a trademark or registered trademark of SUSE LLC or its subsidiaries in Canada, United States and/or other countries.

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

All other trademarks are trademarks of their respective owners.