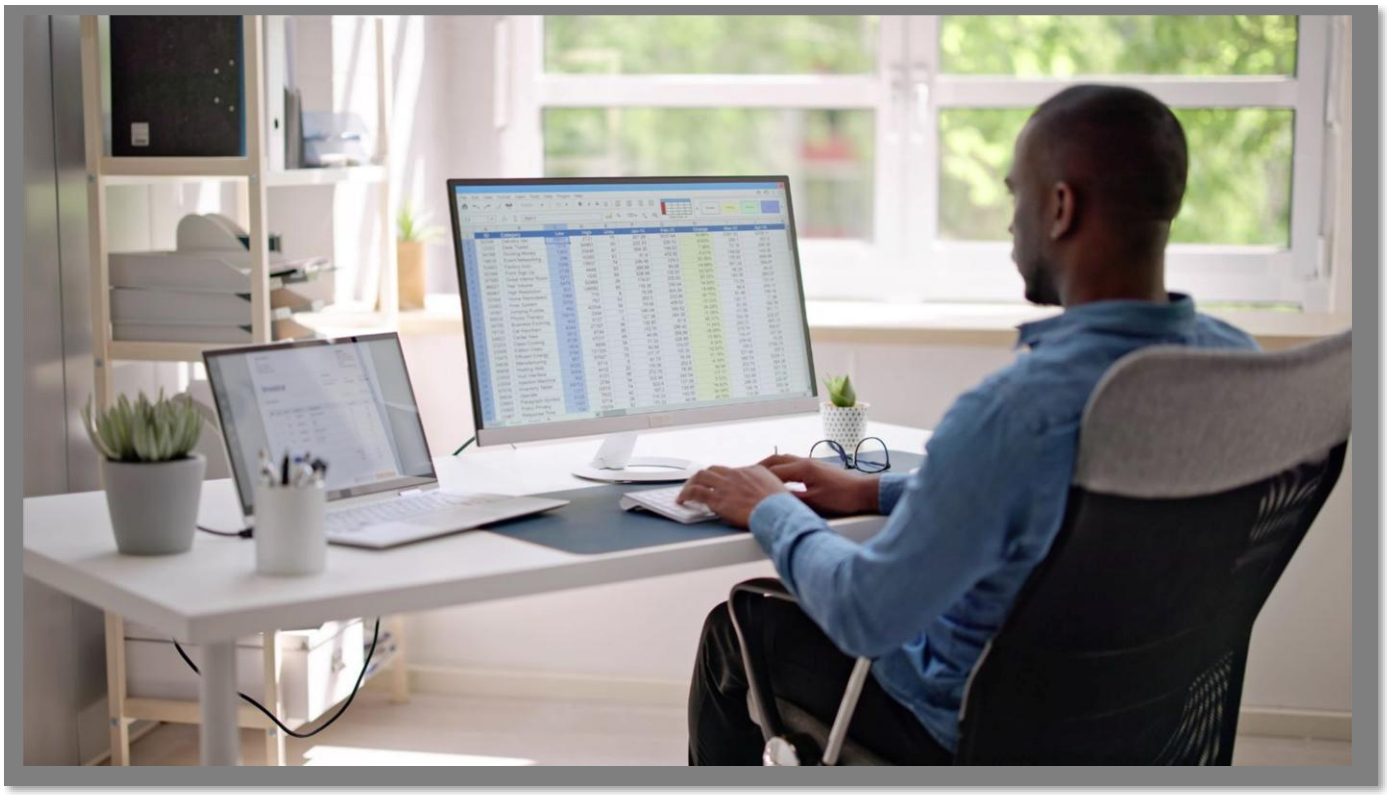
Arbutus Connectors

REST (Representational State Transfer)

CONFIGURATION GUIDE



ARBUTUS
Powerful Analytics Simplified

Contents

Arbutus Connector – REST

A. Introduction

The purpose of this Guide is to provide assistance with configuring the Arbutus REST Connector using the ODBC Data Source Administrator. The configuration process can involve several technical steps that require a good understanding of IT systems and database management.

To make the most of this guide, it's advisable to have a good understanding of database connectivity, driver installation, and system settings. The ODBC Data Source Administrator, which is used as part of the configuration process, allows for the setup and management of data sources, enabling applications to access data from various database systems.

Due to the complexity and potential impact of these configurations, it is recommended that only those individuals with IT or database expertise undertake this task. In addition, it should also be understood that each client's network environment is different. A one-size-fits-all approach is rarely effective, as what works well in one environment may not be suitable in another.

B. About REST

REST (Representational State Transfer) is a way to design web services that allows different systems to communicate over the internet. It uses standard HTTP methods like GET (to retrieve data), POST (to send data), PUT (to update data), and DELETE (to remove data). Each piece of data, called a resource, is identified by a unique URL. REST is popular because it's easy to use, flexible, and works well with the web. It makes it a popular choice for web services and APIs.

To provide some context in terms of how a RESTful API might work, consider that you have a website that shows information about books.

Let's say you have a website that shows information about books. Here's how a RESTful API might work:

1. **GET Request:** If you want to see a list of all books, you would send a GET request to http://example.com/api/books. The server responds with a list of books in JSON format.
2. **POST Request:** To add a new book, you would send a POST request to http://example.com/api/books with the book's details in the request body. The server adds the book and returns a confirmation message.
3. **PUT Request:** If you need to update the details of a specific book, you would send a PUT request to http://example.com/api/books/1 (assuming the book's ID is 1) with the updated information. The server updates the book and returns the updated book details.
4. **DELETE Request:** To delete a book, you would send a DELETE request to http://example.com/api/books/1. The server removes the book and returns a confirmation message.

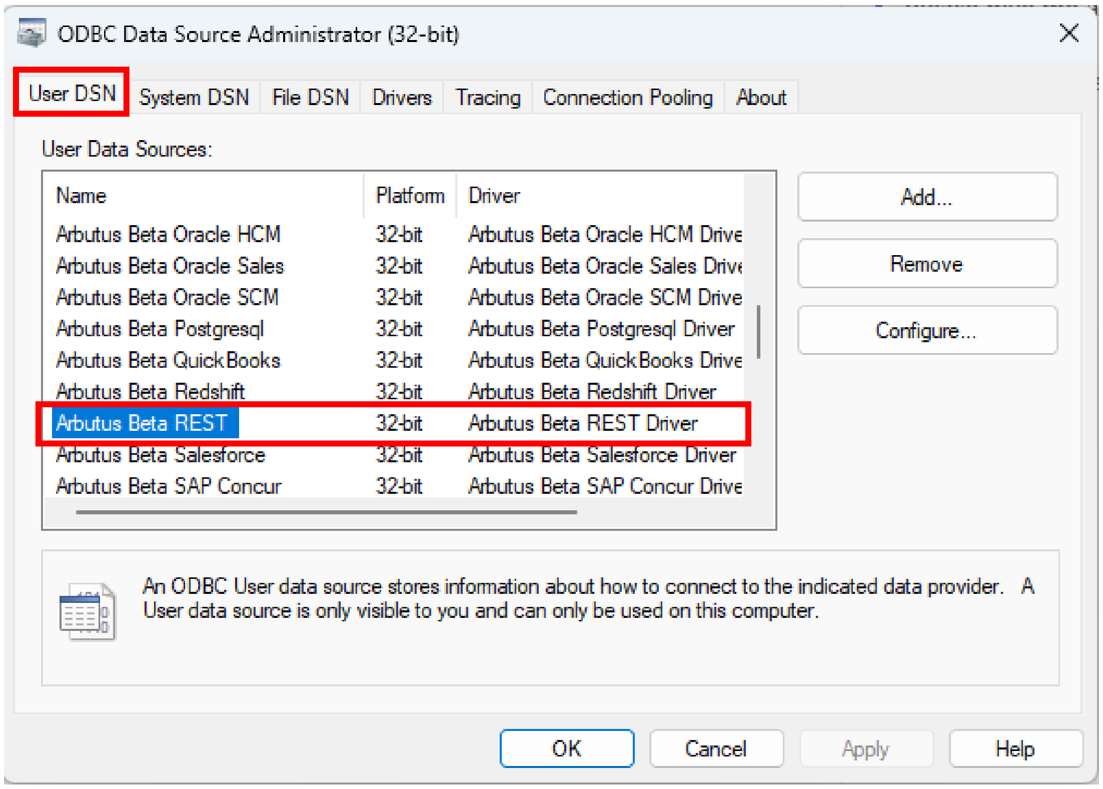
This way, RESTful APIs allow different parts of a web application to communicate and manage resources efficiently.

C. Determining if the Connector exists prior to configuring

Installation of the Arbutus REST Connector is done at the time of installing the Arbutus software. For more information on this, please see the **Overview Guide Document**.

Once the Connector has been installed, the next step is to configure it.

Prior to configuring it, you can check to see if the Connector has been installed by opening the **32-bit ODBC Data Source Administrator**, pictured below, and clicking the **User DSN** tab. Included below is information on how you can access the **ODBC Data Source Administrator**.



User Data Sources:

Name	Platform	Driver
Arbutus Beta Oracle HCM	32-bit	Arbutus Beta Oracle HCM Drive
Arbutus Beta Oracle Sales	32-bit	Arbutus Beta Oracle Sales Drive
Arbutus Beta Oracle SCM	32-bit	Arbutus Beta Oracle SCM Drive
Arbutus Beta Postgresql	32-bit	Arbutus Beta Postgresql Driver
Arbutus Beta QuickBooks	32-bit	Arbutus Beta QuickBooks Drive
Arbutus Beta Redshift	32-bit	Arbutus Beta Redshift Driver
Arbutus Beta REST	32-bit	Arbutus Beta REST Driver
Arbutus Beta Salesforce	32-bit	Arbutus Beta Salesforce Driver
Arbutus Beta SAP Concur	32-bit	Arbutus Beta SAP Concur Drive

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you and can only be used on this computer.

- If the Arbutus REST Connector appears in the list, it can be considered as installed.
- If it is not listed, it is likely that you did not select it during the installation or modification of the Arbutus software. In this case, it is recommended to reinstall the Arbutus software and choose the **Modify** option when prompted. For more details, please refer to the **Overview Guide Document**.

Below is the file path to access and run the **ODBC Data Source Administrator** application:

C:\Windows\SysWOW64\odbcad32.exe

Alternative, you can also try locating and opening the **ODBC Data Source Administrator** application by doing a search on your desktop application.

D. Configuring the Connector after it has been installed

Once you have verified that the Arbutus Connector has been installed, it is time to configure it.

This process is done using the **ODBC Data Source Administrator**. It can be described as "**editing the DSN configuration**".

DSN, Drivers, and Data Sources

What is a DSN? DSN stands for Data Source Name, and is a unique name used to create a data connection to a database using open database connectivity (ODBC).

A DSN is a data structure that contains the information required to connect to a database. It is essentially a string that identifies the source database, including the driver details, the database name, and often authentication credentials and other necessary connection parameters. DSNs facilitate a standardized method for applications to access databases without needing hard-coded connection details, enhancing flexibility and scalability in database management.

- ***Drivers*** are the components that process ODBC requests and return data to the application. If necessary, drivers modify an application's request into a form that is understood by the data source. The **Drivers** tab in the **ODBC Data Source Administrator** dialog box lists all drivers installed on your computer, including the name, version, company, file name, and file creation date of each driver.

- ***Data sources*** are the databases of files accessed by a driver and are identified by a data source name (DSN). You use the ODBC Data Source Administrator to add, configure, and delete data sources from your system.

All ODBC connections require that a DSN be configured to support the connection. When a client application wants to access an ODBC-compliant database, it references the database using the DSN.

The types of DSNs are:

- **User DSN** – User DSNs are local to a computer and can be used only by the current user. They are registered in the HKEY_Current_USER registry subtree.

- **System DSN** – System DSNs are local to a computer rather than dedicated to a user. The system or any user with privileges can use a data source set up with a system DSN. System DSNs are registered in the HKEY_LOCAL_MACHINE registry subtree.

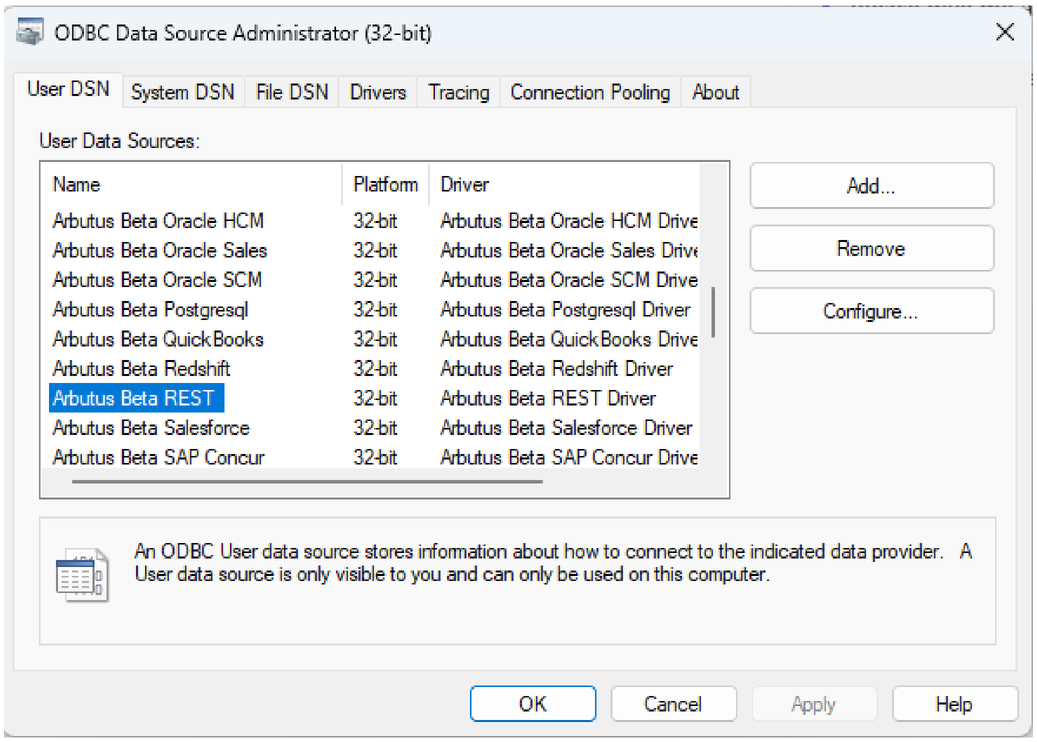
- **File DSN** – File DSNs are file-based sources that can be shared among all users who have the same drivers installed and therefore have access to the database. These data sources need not be dedicated to a user nor be local to a computer. File data source names are identified by a file name with a .dsn extension.

User and system data sources are collectively known as *machine* data sources because they are local to a computer.

Each of these DSNs has a tab in the **ODBC Data Source Administrator** dialog.

Follow these steps to edit the DSN configuration and configure the Connector.

1. First open the **ODBC Data Source Administrator**.



Name	Platform	Driver
Arbutus Beta Oracle HCM	32-bit	Arbutus Beta Oracle HCM Drive
Arbutus Beta Oracle Sales	32-bit	Arbutus Beta Oracle Sales Drive
Arbutus Beta Oracle SCM	32-bit	Arbutus Beta Oracle SCM Drive
Arbutus Beta Postgresql	32-bit	Arbutus Beta Postgresql Driver
Arbutus Beta QuickBooks	32-bit	Arbutus Beta QuickBooks Drive
Arbutus Beta Redshift	32-bit	Arbutus Beta Redshift Driver
Arbutus Beta REST	32-bit	Arbutus Beta REST Driver
Arbutus Beta Salesforce	32-bit	Arbutus Beta Salesforce Driver
Arbutus Beta SAP Concur	32-bit	Arbutus Beta SAP Concur Drive

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you and can only be used on this computer.

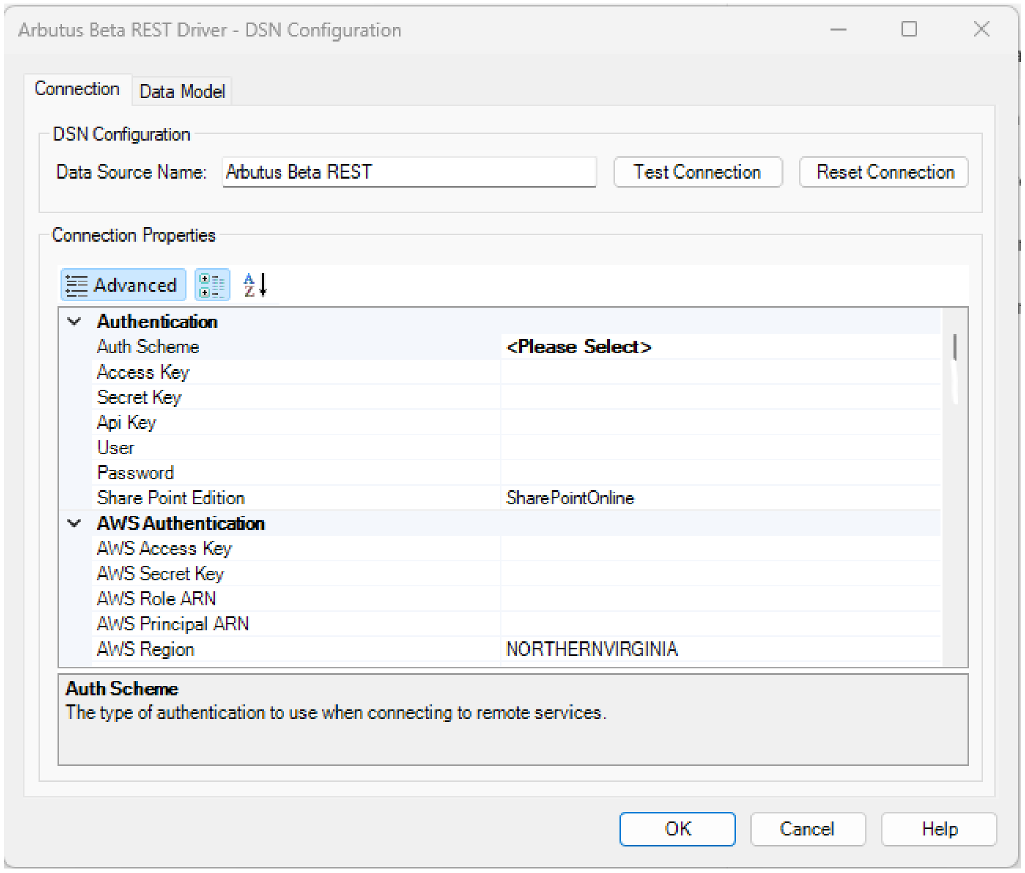
2. Click the **User DSN** tab.

   Selected data connectors are installed as **User DSN's** in Window's 32 Bit **ODBC Data Source Administrator**.

   Also, each of the data connector's names is prefaced with Arbutus, for example, **Arbutus REST.**

3. Select the Arbutus Connector, in this case it is **Arbutus REST**.
4. Click **Configure**.

   This opens the **Arbutus REST Driver – DSN Configuration** dialog.



Data Source Name: Arbutus Beta REST

Authentication	
Auth Scheme	**<Please Select>**
Access Key	
Secret Key	
Api Key	
User	
Password	
Share Point Edition	SharePointOnline
AWS Authentication	
AWS Access Key	
AWS Secret Key	
AWS Role ARN	
AWS Principal ARN	
AWS Region	NORTHERNVIRGINIA

Auth Scheme
The type of authentication to use when connecting to remote services.

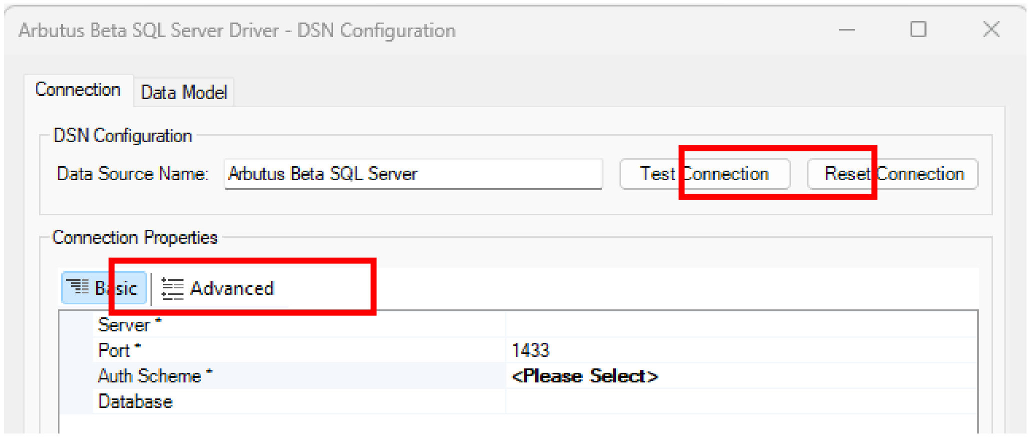
E. Editing the DSN settings – the Advanced tab

With the DSN Configuration dialog open, the next step is to edit the properties, where necessary, in the **Advanced** tab. For example, editing the **SharePointOnline** entry for the **Share Point Edition** property.

E1. Editing the DSN properties in the Advanced tab

This tab includes more detailed and technical settings. It is intended for those users who need more control over the configuration and are comfortable with more complex options. The **Advanced** tab often includes settings that can fine-tune the behaviour of the system feature.

Upon completing the edits in the **Advanced** tab, you can go ahead and try testing the connection via REST by clicking the **Test Connection** button, as highlighted in the screenshot below.



Data Source Name: Arbutus Beta SQL Server

Server *	
Port *	1433
Auth Scheme *	<Please Select>
Database	

In the **Advanced** tab, there are **seventeen** main properties to review:

1. Authentication
2. Azure Authentication
3. Caching
4. Connection
5. CSV
6. Firewall
7. JSON and XML
8. JWT OAuth
9. Kerberos
10. Logging
11. Miscellaneous
12. OAuth
13. Proxy
14. Schema
15. SSH
16. SSL
17. SSO

Each of the above provides a complete list of their respective properties you can configure in the connection settings, as described below.

1. Connection Property: Authentication

This property has the following settings available for configuration:

- Auth Scheme – specify, be selecting from the dropdown list, the type of authentication to use when connecting to remote services. The following options are available for selection:
  - AwsRootKeys
  - AwsECSRoles
  - AwsIAMRoles
  - ADFS
  - Okta
  - PingFederate
  - AwsMFA
  - AwsTempCredentials
  - AwsCredentialFile
  - AzureAD
  - AzureMSI
  - AzureServicePrincipal
  - HMAC
  - OAuth
  - Basic
  - OneLogin
  - SFTP
  - None
  - Negotiate
  - OAuthJWT
  - GCPInstanceAccount
  - Digest
  - OAuthPassword
  - OAuthClient

- AzureServicePrincipalCert
- AccessKey
- OAuthPKCE
- AzureStorageSAS

If necessary, more information on this setting and its properties can be provided.

- Access Key – this is your account access key. This value is accessible from your security credentials page depending on the service you are using.

- Secret Key – this is you account secret key. This value is accessible from your security credentials page depending on the service you are using.

- API Key – this is the API Key used to identify the user to IBM Cloud.

  Access to resources in the REST REST API is governed by an API key in order to retrieve token. An API Key can be created by navigating to Manage > Access (IAM) > Users and clicking 'Create'.

- User – this is the user account used to authenticate. Together with **Password** (see below) this field is used to authenticate against the server.

- Password – this is the password used to authenticate the user. The **User** (see above) and **Password** are together used to authenticate with the server.

- Share Point Edition – select from the dropdown list the edition of SharePoint being used. The options available are:
  - SharePointOnline
  - SharePointOnPremise

  The default value is **SharePointOnline**

2. Connection Property: AWS Authentication

This property has the following settings available for configuration:

- AWS Access Key – this is your AWS account access key. This value is accessible from your AWS security credentials page.

  a. Sign into the **AWS Management** console with the credentials for your root account.
  b. Select your account name or number and select **My Security Credentials** in the menu that is displayed.
  c. Click **Continue to Security Credentials** and expand the **Access Keys** section to manage or create root account access keys.

- AWS Secret Key - this is your AWS account secret key. This value is accessible from your AWS security credentials page.

  See above for steps on how to access this value.

- AWS Role ARN – this is the Amazon Resource Name of the role to use when authenticating.

  When authenticating outside of AWS, it is common to use a Role for authentication instead of your direct AWS account credentials. Entering the **AWS Role ARN** will cause the ODBC Driver for REST to perform a role based authentication instead of using the **AWS Access Key** (see above) and **AWS Secret Key** (see above) directly.

  The **AWS Access Key** and **AWS Secret Key** must still be specified to perform this authentication. You cannot use the credentials of an AWS root user when setting **Role ARN**. The **AWS Access Key** and **AWS Secret Key** must be those of an IAM user.

- AWS Principal ARN – this is the ARN of the SAML Identity provider in your AWS account.

- AWS Region – select from the dropdown list the hosting region for your Amazon Web Services.

  The possible values listed include OHIO, NORTHERNVIRGINIA, NORTHERNCALIFORNIA, OREGON, CAPETOWN, HONGKONG, HYDERABAD, JAKARTA, MALAYSIA, MELBOURNE, MUMBAI, OSAKA, SEOUL, SINGAPORE, SYDNEY, TOKYO, CENTRAL, CALGARY, BEIJING, NINGXIA, FRANKFURT, IRELAND, LONDON, MILAN, PARIS, SPAIN, STOCKHOLM, ZURICH, TELAVIV, BAHRAIN, UAE, SAOPAULO, GOVCLOUDEAST, GOVCLOUDWEST, ISOLATEDUSEAST, ISOLATEDUSEASTB, ISOLATEDUSWEST, ISOLATEDEUWEST

  A default value, e.g., NORTHERNVIRGINIA, may already be listed. You should change this value as appropriate and required.

- AWS Credentials File – this is the path to the AWS Credentials File to be used for authentication.

  See *https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html* for more information.

- AWS Credentials File Profile – this is the name of the profile to be used from the supplied AWS Credentials File.

  You may notice a default value of **default**.

  See *https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html* for more information.

- AWS Session Token – this is your AWS token.

  See Use temporary credentials with AWS resources - AWS Identity and Access Management for more information.

- AWS External ID – this is a unique identifier that might be required when you assume a role in another account.

- MFA Serial Number – this is the serial number of the MFA device, if one is being used.

  You can find the device for an IAM user by going to the AWS Management Console and viewing the user's security credentials. For virtual devices, this is actually an Amazon Resource Name (such as arn:aws:iam::123456789012:mfa/user).

- MFA Token – this is the temporary token available from your MFA device.

  If MFA is required, this value will be used along with the **MFA Serial Number** to retrieve temporary credentials to login. The temporary credentials available from AWS will only last up to 1 hour by default (see below, Temporary Token Duration). Once the time is up, the connection must be updated to specify a new MFA token so that new credentials may be obtained.

- Credentials Location – specify the location of the settings file where MFA credentials are saved.

While there may be a default value already showing for this setting, you can click the **three dots** in the input box, which will then open a dialog allowing you to browse to the desired location.

MFA credentials are short-lived and typically expire after an hour. At that point, you must specify a different **MFA Token** (see above) to continue connecting. MFA tokens only work once, so in the case of multi-threaded or multi-process applications, the credentials must be centrally located and shared to avoid problems. When **MFA Token** is not specified, this property does nothing.

- Temporary Token Duration – specify the amount of time (in seconds) a temporary token will last.

  The default value is **3600** seconds.

  Temporary tokens are used with both MFA and Role based authentication. Temporary tokens will eventually time out, at which time a new temporary token must be obtained. For situations where MFA is not used, this is not a big deal. The ODBC Driver for REST will internally request a new temporary token once the temporary token has expired.

  However, for MFA required connection, a new **MFA Token** must be specified in the connection to retrieve a new temporary token. This is a more intrusive issue since it requires an update to the connection by the user. The maximum and minimum that can be specified will depend largely on the connection being used.

  For Role based authentication, the minimum duration is 900 seconds (15 minutes) while the maximum if 3600 (1 hour). Even if MFA is used with role-based authentication, 3600 is still the maximum.

  For MFA authentication by itself (using an IAM User or root user), the minimum is 900 seconds (15 minutes), the maximum is 129600 (36 hours).

- Server Side Encryption – select from the dropdown list (see below) the appropriate value – when activated, file uploads into Amazon S3 buckets will be server-side encrypted.

  The possible options available for selection are:
  - OFF
  - S3-Managed Keys
  - Key Management Service Keys

  The default value is **OFF.**

  Server-side encryption is the encryption of data at its destination by the application or service that receives it. Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers and decrypts it for you when you access it.

  See *https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html* for more information.

- SSE Context – this is a BASE64-encoded UTF-8 string holding JSON, which represents a string-string (key-value) map.

- SSE Enable S3 Bucket Keys – this is a True/False selection. Select the appropriate value, based on following determination:
  - configuration to use an S3 Buckey Key at the object level when encrypting data with AWS KMS. Enabling this will reduce the cost of server-side encryption by lowering calls to AWS KMS.

  The default value is **False**.

- SSE Key – this is a symmetric Key Management Service Key that is used to protect the data when using **Server Side Encryption** (see above).

3. Connection Property: Azure Authentication

This property has the following settings available for configuration:

- Azure Storage Account – this is the name of your Azure storage account.

- Azure Access Key – this is the storage key associated with your Azure account.

  You can retrieve this access key as follows:

  a. Sign into the azure portal with the credentials for your root account. (https://portal.azure.com/)
  b. Click on **Storage accounts** and select the storage account you want to use.
  c. Under settings, click **Access keys**.
  d. Your storage account name and key will be displayed on that page.

- Azure Shared Access Signature – this is the shared access key signature that may be used for authentication.

  You can create one as follows:

  a. Sign into the azure portal with the credentials for your root account. (https://portal.azure.com/)
  b. Click on **Storage accounts** and select the storage account you want to use.
  c. Under **Settings**, click **Shared Access Signature**.
  d. Set the permissions and when the token will expire
  e. Click **Generate SAS** and you can copy the token.

- Azure Tenant – this is the Microsoft Online tenant being used to access data. If not specified, your default tenant is used.

  For instance, contoso.onmicrosoft.com. Alternatively, specify the tenant Id. This value is the directory Id in the Azure Portal > Azure Active Directory > Properties.

  Typically, it is not necessary to specify the Tenant. This can be automatically determined by Microsoft when using the **OAuth Grant Type** set to CODE (default). However, it may fail in the case that the user belongs to multiple tenants. For instance, if an Admin of domain A invites a user of domain B to be a guest user. The user will now belong to both tenants. It is a good practice to specify the Tenant, although in general things should normally work without having to specify it.

The **Azure Tenant** is required when setting **OAuth Grant Type** (see section '**m**' below) to CLIENT. When using client credentials, there is no user context. The credentials are taken from the context of the app itself. While Microsoft still allows client credentials to be obtained without specifying which Tenant, it has a much lower probability of picking the specific tenant you want to work with. For this reason, we require **Azure Tenant** to be explicitly stated for all client credentials connections to ensure you get credentials that are applicable for the domain you intend to connect to.

- Azure Environment – select from the dropdown list (see below) the Azure Environment to use when establishing a connection.

  The possible options available for selection are:
  - GLOBAL
  - CHINA
  - USGOVT
  - USGOVTDOD

  The default value is **GLOBAL**.

  In most cases, leaving the environment set to global will work. However, if your Azure Account has been added to a different environment, the **Azure Environment** may be used to specify which environment.

4. Connection Property: Caching

This property has the following settings available for configuration:

- **Auth Cache** - this is a True/False selection. Select the appropriate value, based on the following determination:
  - automatically caches the results of SELECT queries into a cache database specified by either Cache Location (see below) or both of **Cache Connection** (see below) and **Cache Provider** (see below).

  When **Auto Cache** = true, the driver automatically maintains a cache of your table's data in the database of your choice.

  The default value is **False**.

  If required, more information on this setting and its properties can be provided.

- Cache Provider – this is the name of the provider to use to cache data.

  You can cache to any database for which you have an ADO.NET provider. The caching database is determined based on the **Cache Provider** and **Cache Connection** (see below) properties.

  If required, more information on this setting and its properties can be provided.

- Cache Driver – this is the database driver used to cache data.

- Cache Connection – this is the connection string for the cache database.

  This property is always used in conjunction with **Cache Provider** (see above). Setting both properties will override the value set for **Cache Location** (see below) for caching data.

- Cache Location – specify the path to the cache when caching to a file.

  While there may be a default value already showing for this setting, you can click the **three dots** in the input box, which will then open a dialog allowing you to browse to the desired location.

- Cache Tolerance – this is the tolerance for stale data in the cache specified in seconds when using **Auto Cache** (see above).

  This only applies when **Auto Cache** is used. The driver checks with the data source for newer records after the tolerance interval has expired. Otherwise, it returns the data directly from the cache.

  The default value is **600** seconds.

- Offline - this is a True/False selection. Select the appropriate value, based on the following determination:
  - use offline mode to get the data from the cache instead of the live source.

  When **Offline** = true, all queries execute against the cache as opposed to the live data source. In this mode, certain queries like INSERT, UPDATE, DELETE, and CACHE are not allowed.

  The default value is **False**.

- Cache Metadata - this is a True/False selection. Select the appropriate value, based on the following determination:
  - this property determines whether or not to cache the table metadata to a file store.

  As you execute queries with this property set, table metadata in the REST catalog is cached to the file store specified by **Cache Location** (see above) if set or the user's home directory otherwise. A table's metadata is retrieved only once, when the table is queried for the first time.

  Cache Metadata becomes useful when metadata operations are expensive, such as when you are working with large amounts of metadata or when you have many short-lived connections

  The default value is **False**.

5. Connection Property: Connection

This property has the following settings available for configuration:

- Use Lake Formation – this is a True/False selection. Select the appropriate value, based on the following determination:
  - when this property is set to True, AWS Lake Formation service will be used to retrieve temporary credentials, which enforces access policies against the user based on the configured IAM role.

  The service can be used when authenticating through OKTA, ADFS, AzureAD, PingFederate, while providing a SAML assertion.

  The default value is **False**.

- Connection Type – specify, by selecting from the dropdown list, the file storage service of server, or file access protocol through which your REST files are stored and retrieved. The following options are available for selection:
  - Auto
  - Local
  - Amazon S3
  - Azure Blob Storage
  - Azure Data Lake Storage Gen 1
  - Azure Data Lake Storage Gen 2
  - Azure Data Lake Storage Gen 2 SSL
  - Azure Files
  - Box
  - Dropbox
  - FTP
  - FTPS
  - Google Cloud Storage
  - Google Drive
  - HDFS
  - HDFS Secure
  - HTTP
  - HTTPS
  - IBM Object Storage Source
  - OneDrive
  - Oracle Cloud Storage
  - SFTP
  - SharePoint REST
  - SharePoint SOAP

  The default value is **Auto.**

- Format – select from the dropdown list the format reported by the data source. The options available are:
  - JSON
  - XML
  - CSV

  The default value is **JSON**.

- URI – this is the Uniform Resource Identifier (URI) for the XML/JSON/CSV resource location.

  If required, more information on this setting and its properties can be provided.

- Region – this is the hosting region for your S3-like Web Services. For example, **ap-melbourne-1** for the **Australia Southeast (Melbourne)** Region.

  If required, more information on the different Commercial Cloud Regions can be provided.

- Project Id – this is the Id of the project where your Google Cloud Storage instance resides.

  You can find this value by going to Google Cloud Console and clicking the project name at the top left screen. The **Project Id** is displayed on the Id column of the matching project.

- Oracle Namespace – this is the Oracle Cloud Object Storage namespace to use.

  This setting must be set to the Oracle Cloud Object Storage namespace associated with the Oracle Cloud account before any requests can be made.

  For more information, refer to the Understanding Object Storage Namespaces page of the Oracle Cloud documentation for instructions on how to find your account's Object Storage namespace.

- Storage Base URL – this is the URL of a cloud storage service provider.

  This connection property is used to specify:

  - The URL of a custom S3 service
  - The URL required for the SharePoint SOAP/REST cloud storage service provider.

    If the domain for this option ends in **-my** (for example, ***https://bigcorp-my.sharepoint.com***) then you may need to use the onedrive:// scheme instead of the sp:// or sprest:// scheme.

- Simple Upload Limit – this setting specifies the threshold, in bytes, above which the provider will choose to perform a multipart upload rather than uploading everything in one request.

- Use Virtual Hosting – this is a True/False selection. Select the appropriate value, based on the following determination:
  - If True (default), buckets will be referenced in the request using the hosted-style request: *http://yourbucket.s3.amazonaws.com/your object*. If set to False, the bean will use the path-style request: *http://s3.amazonaws.com/yourbucket/yourobject*.

  Note that this property will be set to False, in case of an S3 based custom service when the **Custom URL** is specified.

  The default value is **True.**

6. Connection Property: CSV

This property has the following settings available for configuration:

- FMT – specify the format that will be used to parse all CSV files. When this connection property is set, the driver parses all CSV files in the URI according to the specified file format.

  You can set the FMT to one of the following values:
  - CsvDelimited: The driver separates each field by commas.
  - TabDelimited: The driver separates each field by tabs.
  - FixedLength: The driver counts a specified number of characters to separate each field. If this format is specified, the width of each column must be defined in Schema.ini.

  The default value is **CsvDelimited**.

- Include Column Headers - this is a True/False selection. Select the appropriate value, based on the following determination:
  - Whether to get column names from the first line of CSV files.

  When this property is set to **True**, the driver will derive column names for each table from the first row of each file. When this property is set to False, column names are simply the column numbers.

  The default value is **True**.

7. Connection Property: Firewall

This property has the following settings available for configuration:

- Firewall Type – select from the dropdown list the protocol used by a proxy-based firewall. The options available are:
  - NONE
  - TUNNEL
  - SOCKS4
  - SOCKS5

  This property specifies the protocol that the driver will use to tunnel traffic through the **Firewall Server** (see below) proxy. Note that by default, the driver connects to the system proxy; to disable this behavior and connect to one of the following proxy types, set **Proxy Auto Detect** to False.

  The default value is **NONE**.

  If required, more information on this property, e.g., Port settings, can be provided.

- Firewall Server – specifies the IP address, DNS name, or host name of a proxy allowing traversal of a firewall. The protocol is specified by **Firewall Type** (see above). Use **Firewall Server** with this property to connect through SOCKS or do tunneling. Use **Proxy Server** to connect to an HTTP proxy.

  Note that the driver uses the system proxy by default. To use a different proxy, set **Proxy Auto Detect** to false.

- Firewall Port – this is the TCP port for a proxy-based firewall.

  Use **Firewall Server** to specify the name or IP address. Specify the protocol with **Firewall Type**.

  The default value is **0**.

- Firewall User – this is the username to use to authenticate with a proxy-based firewall.

  The **Firewall User** and **Firewall Password** (see below) properties are used to authenticate against the proxy specified in **Firewall Server** (see above) and **Firewall Port** (see above), following the authentication method specified in **Firewall Type** (see above).

- Firewall Password - this is a password to use to authenticate with a proxy-based firewall

  This property is passed to the proxy specified by **Firewall Server** (see above) and **Firewall Port** (see above), following the authentication method specified by **Firewall Type** (see above).

8. Connection Property: JSON and XML

This property has the following settings available for configuration:

- Backwards Compatible Mode – this is a True/False selection. Select the appropriate value, based on the following determination:
  - Set Backwards Compatibility Mode to True to use the XML/JSON functionality and features available in the 2017 version.

  The default value is **False**.

- Qualify Columns – select from the dropdown list what controls whether the provider will use relative column names. The options available are:
  - none
  - parent
  - full

  By default, the driver will only qualify a column name as much as is necessary to make it unique.

  The default value is **none**.

  If required, more information on this setting and its properties can be provided.

- URI Separator – this is a delimiter used to separate different values in the URI property.

  By default, the delimiter is a comma which means that multiple URIs can be joined together. For example:

  URI=c:/data/json1.json,c:/data/json2.json,c:/data/json3.json

  The default value is a **comma**.

- XPath – this is the XPath of an element that repeats at the same height within the XML/JSON document (used to split the document into multiple rows).

  Multiple paths can be specified using a semi-colon separated list. **Data Model** allows you to configure how the XPath values will be used to create tables and display data.

If left empty, the ODBC Driver for REST will determine the XPaths by parsing the REST document and identifying the object arrays.

This property will be used to generate the schema definition when a schema file (RSD) file is not present.

- Data Model – select from the dropdown list the data model to use when parsing XML/JSON documents and generating the database, metadata. The options available are:
  - Documents
  - FlattenedDocuments
  - Relational

  The driver splits JSON documents into rows based on the objects nested in arrays. Select a **Data Model** configuration to configure how the driver models nested object arrays into tables.

  The default value is **Document**.
  If required, more information on this setting and its properties can be provided.

- JSON Format – select from the dropdown list the format of the JSON document which enables parsing specifically for the selected format. Only has an effect when Format is set to JSON. The options available are:
  - JSON
  - JSONRows
  - LDJSON

  The default value is **JSON**.

  If required, more information on this setting and its properties can be provided.

- XML Format – select from the dropdown list the format of the XML document. The options available are:
  - XML
  - XMLTABLE

  The default value is **XML**.

  If required, more information on this setting and its properties can be provided.

- Flatten Arrays – By default, nested arrays are returned as strings of XML/JSON. The **Flatten Arrays** property can be used to flatten the elements of nested arrays into columns of their own. This is only recommended for arrays that are expected to be short.

  Set **Flatten Arrays** to the number of elements you want to return from nested arrays. The specified elements are returned as columns. The zero-based index is concatenated to the column name. Other elements are ignored.

  For example, you can return an arbitrary number of elements from an array of strings:

  ["FLOW-MATIC","LISP","COBOL"]

  When **Flatten Arrays** is set to 1, the preceding array is flattened into the following table:

Column Name	Column Value
Languages.0	FLOW-MATIC

  Setting **<u>Flatten Arrays</u>** to -1 will flatten all the elements of nested arrays.

- Flatten Objects – Set **Flatten Objects** to true to flatten object properties into columns of their own. Otherwise, objects nested in arrays are returned as strings of XML/JSON. To generate the column name, the driver concatenates the property name onto the object name with a dot.

  The default value is **True**.

9. Connection Property: JWT OAuth

This property has the following settings available for configuration:

- OAuth JWT Audience – this is a space-separated list of entities that may use the JWT. The entries in this list are typically URLs, but the exact values depend on the API being used.

- OAuth JWT Encryption – select from the dropdown list the encryption algorithm to be used in JWT authentication. The options available are:
  - RS256
  - RS384
  - RS512
  - ES256
  - ES284
  - ES512

  The default value is **RS256**.

- OAuth JWT Headers – this is a collection of extra headers that should be included in the JWT headers. Example: header1 = value1, header2 = value2.

- OAuth JWT Validity Time – specify how long the JWT should remain valid, in seconds.

  By default, this is set to 3600 which means the JWT is valid for 1 hour after it is generated. Some APIs may require lower values than this.

  The default value is **3600** seconds.

- OAuth JWT Cert – this is the JWT Certificate store - the name of the certificate store for the client certificate.

  The **OAuth JWT Cert Type** (see below) field specifies the type of the certificate store specified by **OAuth JWT Cert**. If the store is password protected, specify the password in **OAuth JWT Cert Password** (see below)

  OAuth JWT Cert is used in conjunction with the **OAuth JWT Cert Subject** (see below) field in order to specify client certificates.

  If **OAuth JWT Cert** has a value, and **OAuth JWT Cert Subject** is set, a search for a certificate is initiated. Please refer to the **OAuth JWT Cert Subject** field for details.

  If required, more information on this setting and its properties can be provided.

- OAuth JWT Cert Type - select from the dropdown list the type of key store containing the JWT Certificate. The options available are:

  - USER
  - MACHINE
  - PFXFILE
  - PFXBLOB
  - JKSFILE
  - JKSBLOB
  - P7BFILE
  - PPKFILE
  - GOOGLEJSON
  - PEMKEY_FILE
  - PEMKEY_BLOB
  - PUBLIC_KEY_FILE
  - PUBLIC_KEY_BLOB
  - SSHPUBLIC_KEY_FILE
  - SSHPUBLIC_KEY_BLOB
  - XMLFILE
  - XMLBLOB
  - GOOGLEJSONBLOB

  The default value is **USER**.

  If required, more information on this setting and its properties can be provided.

- OAuth JWT Cert Password – select the password for the OAuth JWT certificate.

  If the certificate store is of a type that requires a password, this property is used to specify that password in order to open the certificate store.

  This is not required when using the GOOGLEJSON **OAuth JWT Cert Type.** Google JSON keys are not encrypted.

- OAuth JWT Cert Subject – this is the subject of the OAuth JWT certificate.

  When loading a certificate, the subject is used to locate the certificate in the store.

  If an exact match is not found, the store is searched for subjects containing the value of the property.

  If a match is still not found, the property is set to an empty string, and no certificate is selected.

  The certificate subject is a comma separated list of distinguished name fields and values. For instance, "CN=www.server.com, OU=test, C=US, E=support@cdata.com"

  The special value "*" picks the first certificate in the certificate store.

  The default value is *.

- OAuth JWT Issuer – this is the issuer of the Java Web Token. This is typically either the Client Id or Email Address of the OAuth Application.

- OAuth JWT Subject – this is the user subject for which the application is requesting delegated access. This is typically, the user account name or email address.

10. Connection Property: Kerberos

This property has the following settings available for configuration:

- Kerberos KDC – this is the Kerberos Key Distribution Centre (KDC) service used to authenticate the user.

  The Kerberos properties are used when using SPNEGO or Windows Authentication. The driver will request session tickets and temporary session keys from the Kerberos KDC service. The Kerberos KDC service is conventionally collocated with the domain controller.

  If Kerberos KDC is not specified, the driver will attempt to detect these properties automatically from the following locations:
  - KRB5 Config File (krb5.ini/krb5.conf)
  - Domain Name and Host

- Kerberos Realm – this is the Kerberos Realm used to authenticate the user with the Kerberos Key Distribution Service (KDC). The Kerberos Realm can be configured by an administrator to be any string, but conventionally it is based on the domain name.

- Kerberos SPN – this is the service principal name (SPN) for the Kerberos Doman Controller.

  If the SPN on the Kerberos Domain Controller is not the same as the URL that you are authenticating to, use this property to set the SPN.

- Kerberos Keytab File – this is the Keytab file containing your pairs of Kerberos principals and encrypted keys.

- Kerberos Service Realm – this is the Kerberos realm of the service. In most cases, a single realm and KDC machine are used to perform the Kerberos authentication, and this property is not required.

- Kerberos Service KDC – this is the Kerberos KDC of the service when using cross-realm Kerberos authentication. In most cases, a single realm and KDC machine are used to perform the Kerberos authentication, and this property is not required.
- Kerberos Ticket Cache – this is the full path to an MIT Kerberos credential cache file.

11. Connection Property: Logging

This property has the following settings available for configuration:

- Logfile – this is a file path which designates the name and location of the log file.

  You can click the **three dots** in the input box, which will then open a dialog allowing you to browse to the desired location.

  Once this property is set, the driver will populate the log file as it carries out various tasks, such as when authentication is performed, or queries are executed. If the specified file doesn't already exist, it will be created.

  Connection strings and version information are also logged, though connection properties containing sensitive information are masked automatically.

  If a relative file path is supplied, the location of the log file will be resolved based on the path found in the **Location** connection property.

  For more control over what is written to the log file, you can adjust the **Verbosity** property.

Log contents are categorized into several modules. You can show/hide individual modules using the **Log Modules** (see below) property.

To edit the maximum size of a single logfile before a new one is created, see below for **Max Log File Size**.

If you would like to place a cap on the number of logfiles generated, use **Max Log File Count** (see below).

- Verbosity – this is the verbosity level that determines the amount of detail included in the log file. **Verbosity** levels from 1 to 5 are supported

  The default value is **1**.

- Log Modules – this is the core modules to be included in the log file.

  Only the modules specified (separated by ';') will be included in the log file. By default, all modules are included.

- Max Log File Size – this is a string specifying the maximum size in bytes for a log file. For example, 10MB.

  When the limit is hit, a new log is created in the same folder with the date and time appended to the end. The default limit is 100 MB. Values lower than 100 kB will use 100 kB as the value instead.

  Adjust the maximum number of logfiles generated with **Max Log File Count** (see below).

  The default value is **100MB**.

- Max Log File Count – this is a string specifying the maximum file count of log files.

  When the limit is hit, a new log is created in the same folder with the date and time appended to the end and the oldest log file will be deleted.

  The minimum supported value is 2. A value of 0 or a negative value indicates no limit on the count.

  Adjust the maximum size of the logfiles generated with **Max Log File Size** (see above).

  The default value is **-1**.

12. Connection Property: Miscellaneous

This property has the following settings available for configuration:

- Batch Size – this is the maximum size of each batch operation to submit.

  When Batch Size is set to a value greater than 0, the batch operation will split the entire batch into separate batches of size Batch Size. The split batches will then be submitted to the server individually. This is useful when the server has limitations on the size of the request that can be submitted.

  Setting Batch Size to 0 will submit the entire batch as specified.

  The default value is **0**.

- Charset – this specifies the session character set for encoding and decoding character data transferred to and from the REST file.

  The default value is **UTF-8**.

- Client Culture – this property can be used to specify the format of data (e.g., currency values) that is accepted by the client application. This property can be used when the client application does not support the machine's culture settings. For example, Microsoft Access requires 'en-US'.
  This option affects the format of driver output. To specify the format that defines how input should be interpreted, use the Culture option. By default, the driver uses the current locale settings of the machine to interpret input and format output.

- Culture – This setting can be used to specify culture settings that determine how the provider interprets certain data types that are passed into the provider. For example, setting Culture='de-DE' will output German formats even on an American machine.

  This property affects the driver input. To interpret values in a different cultural format, use the Client Culture property. By default, the driver uses the current locale settings of the machine to interpret input and format output.

- Custom Headers – Other headers as determined by the user (optional).

  This property can be set to a string of headers to be appended to the HTTP request headers created from other properties, like **Content Type**, **From**, and so on.

  The headers must be of the format "header: value" as described in the HTTP specifications. Header lines should be separated by the carriage return and line feed (CRLF) characters.

Use this property with caution. If this property contains invalid headers, HTTP requests may fail.

This property is useful for fine-tuning the functionality of the driver to integrate with specialized or nonstandard APIs.

- Custom URL Params – this is the custom query string to be included in the request.

  The **Custom Url Params** allow you to specify custom query string parameters that are included with the HTTP request. The parameters must be encoded as a query string in the form field1=value1&field2=value2&field3=value3. The values in the query string must be URL encoded.

- Data Source – This property specifies a URI for the resource location.

  This property is an alias to the URI property. Below are examples of the URI formats for the available data sources:

Service Provider	**URI Formats**
Local	localPath/file.json file://localPath/file.xml
HTTP or HTTPS	*http://remoteStream* *https://remoteStream*
Amazon S3	s3://remotePath/file.json
Azure Blob Storage	azureblob://mycontainer/myblob
Google Drive	gdrive://remotePath/file.xml
Box	box://remotePath/file.csv
FTP or FTPS	*ftp://server:port/remotePath/file.json* *ftps://server:port/remotepath/file.csv*

- Default Domain – this property is used for the Oracle Database Gateway for ODBC.

  The Oracle Database Gateway will always truncate the username from the "@" character. When **Default Domain** is specified, an "@" character and the domain will be appended to the username before authenticating.

- Directory Retrieval Depth – this is to Limit the subfolders recursively scanned when **Include Subdirectories** is enabled.

  The default value is **-1**.

- Enable Foreign Key Detection – this is to indicate whether to detect the foreign keys in ODBC.

  The default value is **False**.

- Exclude Files – this is a Comma-separated list of file extensions to exclude from the set of the files modeled as tables.

  It is also possible to specify datetime filters. We currently support **Created Date** and **Modified Date**. All extension filters are evaluated in disjunction (using OR operator), and then the resulting filter is evaluated in conjunction (using AND operator) with the datetime filters. For example:

  ExcludeFiles="TXT,CreatedDate <= '2020-11-26T07:39:34-05:00'"

- Folder Id – this is the ID of a folder in Google Drive. If set, the resource location specified by the URI is relative to the Folder ID for all operations.

- Generate Schema Files – select from the dropdown list the appropriate value that indicates the user preference as to when schemas should be generated and saved. The options available are:
  - Never – a schema file will never be generated.
  - OnUser – a schema file will be generated the first time a table is referenced, provided the schema file for the table does not already exist.
  - OnStart - schema file will be generated at connection time for any tables that do not currently have a schema file.
  - OnCreate - schema file will be generated by when running a CREATE TABLE SQL query.

  The default value is **Never**.

  When you set **Generate Schema Files** to **OnUse**, the driver generates schemas as you execute SELECT queries. Schemas are generated for each table referenced in the query.

  When you set **Generate Schema Files** to **OnCreate**, schemas are only generated when a CREATE TABLE query is executed.

  Another way to use this property is to obtain schemas for every table in your database when you connect. To do so, set **Generate Schema Files** to **OnStart** and connect.

  This property outputs schemas to .rsd files in the path specified by **Location**.

- Include Dropbox Team Resources – this is a True/False selection. Select the appropriate value, based on the following determination:
  - This indicates if you want to include Dropbox team files and folders.

  In order to access Dropbox team folders and files, please set this connection property to True.

  The default value is **False.**

- Include Dual Table – this is a True/False selection. Select the appropriate value, based on the following determination:
  - Set this property to mock the Oracle DUAL table for better compatibility with Oracle database.

  This table is used by Oracle database in a few special cases. This property facilitates connectivity when accessing REST as a remote Oracle database through the ODBC Gateway. For example, in SQL Developer, this table is queried to test the connection.

  The default value is **False.**

- Include Files – this is a Comma-separated list of file extensions to include into the set of the files modeled as tables. For example, IncludeFiles=TXT,TAB.

  No default is set for the property. If **Include Files** is not specified, the driver will infer the following based on **Format**:

  - **Format=CSV**: infers equivalent of **IncludeFiles=CSC,TXT,TAB**.
  - **Format=JSON**: infers equivalent of **IncludeFiles=JSON**.
  - **Format=XML**: infers equivalent of **IncludeFiles=XML**.

  A 'NOEXT' value can be specified to include files without an extension.

File masks can be specified using an asterisk (*) to provide enhanced filtering capabilities. For example, **IncludeFiles**=2020*.csv,TXT.

It is also possible to specify datetime filters. We currently support **Created Date** and **Modified Date**. All extension filters are evaluated in disjunction (using OR operator), and then the resulting filter is evaluated in conjunction (using AND operator) with the datetime filters.

- Include Items From All Drives – this is a True/False selection. Select the appropriate value, based on the following determination:
  - Whether Google Drive shared drive items should be included in results. If not present or set to false, then shared drive items are not returned.

  If this property is set to **True**, files will be retrieved from all drives, including shared drives. The file retrieval can be limited a specific shared drive or a specific folder in that shared drive by setting the start of the URI to the path of the shared drive and optionally any folder within, for example: 'gdrive://SharedDriveA/FolderA/...'.

  Additionally, the **Folder Id** property can be used to limit the search to an exact subdirectory.

  The default value is **False.**

- Limit Key Size – this is the maximum length of a primary key column.

  In some ODBC tools, for example, Microsoft Access, the length of the primary key column cannot be larger than a specific value. This property makes the ODBC Driver override the reported length of all the primary key columns. It is especially useful when using the ODBC Driver as a Microsoft Access Linked Data Source.

  Setting the **Limit Key Size** to zero will make the key length revert to the original length

  The default value is **255**.

- Map Bigint TO Varchar – this is a True/False selection. Select the appropriate value, based on the following determination:
  - This property controls whether or not the **bigint** type maps to SQL_VARCHAR instead of SQL_BIGINT.

  The default value is **False.**

- Map To Int – this is a True/False selection. Select the appropriate value, based on the following determination:
  - This property controls whether or not the long type maps to SQL_INTEGER instead of SQL_BIGINT.

  The default value is **False.**

- Map To Long Varchar – this property controls whether or not a column is returned as SQL_LONGVARCHAR.

  Some applications require all text data larger than a certain number of characters to be reported as SQL_LONGVARCHAR. Use **Map To Long Varchar** to map any column larger than the specified size so they are reported as SQL_LONGVARCHAR instead of SQL_VARCHAR.

  The default value is **-1**.

- Map To WVarchar – this is a True/False selection. Select the appropriate value, based on the following determination:
  - This property controls whether or not string types map to SQL_WVARCHAR instead of SQL_VARCHAR.

  String columns must be mapped to SQL_WVARCHAR to accommodate various international character sets, so **Map To WVarchar** is set to true by default. You may set it to false to use SQL_VARCHAR instead.

  The default value is **True.**

- Max Rows – this limits the number of rows returned when no aggregation or GROUP BY is used in the query. LIMIT clauses take precedence over the limit specified in **Max Rows**.

  The default value is **-1**.

- Maximum Column Size – this is the maximum column size.

  Some tools restrain the largest size of a column or the total size of all the columns selected. You can set the **Maximum Column Size** to overcome these schema-based restrictions. The driver will not report any column to be larger than the **Maximum Column Size**.

  Set a **Maximum Column Size** of zero to eliminate limits on column size

  The default value is **16000**.

- Metadata Discovery URI – this is used when aggregating multiple files into one table, this property specifies a specific file to read to determine the aggregated table schema.

- Other - These hidden properties are used only in specific use cases.

  If necessary, more information on this setting and its properties can be provided.

- Pagesize – this is the maximum number of results to return per page from REST.

  The **Page size** property affects the maximum number of results to return per page from REST. While the data source optimizes the default page size for most use cases, you may need to adjust this value depending on the specific object or service endpoint you are querying. Increasing the page size may improve performance, but it could also result in higher memory consumption per page.

  The default value is **1000**.

- Pseudo Columns – specify a set of **pseudo columns** to expose as columns.

  The value of this connection setting is of the format "Table1=Column1;Table1=Column2;Table2=Column3".

  You can use the "*" character to include all tables and all columns; for example, "*=*".

- RTK – this is the runtime key used for licensing.

  The RTK property may be used to license a build. Only use this property in environments that do not support the .NET licensing scheme. In most circumstances, this property is not necessary. The value of this property overwrites all licensing information, so use caution to ensure that the value is correct.

- Readonly – this is a True/False selection. Select the appropriate value, based on the following determination:
  - You can use this property to enforce read-only access to REST from the provider.

  If this property is set to true, the driver will allow only SELECT queries. INSERT, UPDATE, DELETE, and stored procedure queries will cause an error to be thrown.

  The default value is **False.**

- Row Scan Depth – this is the number of rows to scan when dynamically determining columns for the table.

  Higher values will result in a longer request but will be more accurate.

  Setting this value to 0 (zero) will parse the entire document

  The default value is **100**.

- Timeout – this is value in seconds until the timeout error is thrown, canceling the operation.

  If **Time out** = 0, operations do not time out. The operations run until they complete successfully or until they encounter an error condition.

  If **Time out** expires and the operation is not yet complete, the driver throws an exception

  The default value is **60**.

- Type Detection Scheme – select from the dropdown list how to determine the data types of columns. The options available are:
  - None
  - RowScan

  The default value is **RowScan**.

- Upper Case Identifiers – this is a True/False selection. Select the appropriate value, based on the following determination:
  - This property reports all identifiers in uppercase. This is the default for Oracle databases and thus allows better integration with Oracle tools such as the Oracle Database Gateway.

  The default value is **False.**

- User Defined Views – this is a file path pointing to the JSON configuration file containing your custom views.

  User Defined Views are defined in a JSON-formatted configuration file called UserDefinedViews.json. The driver automatically detects the views specified in this file

  You can also have multiple view definitions and control them using the **User Defined Views** connection property. When you use this property, only the specified views are seen by the driver.

13. Connection Property: OAuth

This property has the following settings available for configuration:

- Initiate OAuth – select from the dropdown list the appropriate property to initiate the process to obtain or refresh the OAuth access token when you connect. The options available are:
  - OFF - indicates that the OAuth flow will be handled entirely by the user. An **OAuth Access Token** (see below) will be required to authenticate.
  - REFRESH - indicates that the entire OAuth Flow will be handled by the driver. If no token currently exists, it will be obtained by prompting the user via the browser. If a token exists, it will be refreshed when applicable.
  - GETANDREFRESH - indicates that the driver will only handle refreshing the **OAuth Access Token**. The user will never be prompted by the driver to authenticate via the browser. The user must handle obtaining the **OAuth Access Token** and **OAuth Refresh Token** (see below) initially.

- OAuth Version – select from the dropdown list the version of OAuth being used. The options available are:
  - 1.0
  - 2.0

  The default value is **2.0**.

- OAuth Client Id – this is the client Id assigned when you register your application with an OAuth authorization server.

  As part of registering an OAuth application, you will receive the **OAuth Client Id** value, sometimes also called a consumer key, and a client secret, the **OAuth Client Secret**.

- OAuth Client Secret – this is the client secret assigned when you register your application with an OAuth authorization server.

  As part of registering an OAuth application, you will receive the **OAuth Client Id**, also called a consumer key. You will also receive a client secret, also called a consumer secret. Set the client secret in the **OAuth Client Secret** property.

- OAuth Settings Location – this is the location of the settings file where OAuth values are saved when **Initiate OAuth** is set to GETANDREFRESH or REFRESH . Alternatively, you can hold this location in memory by specifying a value starting with 'memory://'.

  When **Initiate OAuth** is set to **GETANDREFRESH** or **REFRESH**, the driver saves OAuth values to avoid requiring the user to manually enter OAuth connection properties and to allow the credentials to be shared across connections or processes.

  The default value is **registry://%DSN%**.

  If necessary, more information on this setting and its properties can be provided.

- Callback URL – this is the OAuth callback URL to return to when authenticating. During the authentication process, the OAuth authorization server redirects the user to this URL. This value must match the callback URL you specify in your app settings.

- Scope – specify scope to obtain the initial access and refresh token.

- OAuth Grant Type – select from the dropdown list the grant type for the OAuth flow. The options available are:
  - CODE
  - CLIENT
  - PASSWORD

  The default value is **Post**.

- OAuth Password Grant Mode – select from the dropdown list how the OAuth Client Id and Client Secret should be passed. The options available are:
  - Post
  - Basic

  The default value is **CODE**.

- OAuth Include Callback URL – this is a True/False selection. Select the appropriate value, based on the following determination:
  - Whether to include the callback URL in an access token request.

  This option should only be enabled for OAuth services that report errors when **redirect_uri** is included.

  The default value is **True**.

- OAuth Authorization URL – this is the authorization URL for the OAuth service.

  At this URL, the user logs into the server and grants permissions to the application. In OAuth 1.0, if permissions are granted, the request token is authorized.

- OAuth Access Token URL – this is the URL to retrieve the OAuth access token from.

  In OAuth 1.0, the authorized request token is exchanged for the access token at this URL.

- OAuth Refresh Token URL – this is the URL to refresh the OAuth token from.

  In OAuth 2.0, this URL is where the refresh token is exchanged for a new access token when the old access token expires.

- OAuth Request Token URL – this is the URL the service provides to retrieve request tokens from. This is required in OAuth 1.0. In OAuth 1.0, this is the URL where the app makes a request for the request token.

- OAuth Verifier – this is the verifier code returned from the OAuth authorization URL. This can be used on systems where a browser cannot be launched such as headless systems.

- PKCE Verifier – this is the PKCE code verifier generated from executing the **Get OAuth Authorization Url** stored procedure for PKCE authentication schemes.

  The Proof Key for Code Exchange code verifier generated from executing the **Get OAuth Authorization Url** stored procedure for PKCE authentication schemes. This can be used on systems where a browser cannot be launched such as headless systems.

- Auth Token – this is the authentication token used to request and obtain the OAuth Access Token.

  This property is required only when performing headless authentication in OAuth 1.0. It can be obtained from the **Get OAuth Authorization Url** stored procedure.

  It can be supplied alongside the **Auth Key** in the **Get OAuth Access Token** stored procedure to obtain the **OAuth Access Token**.

- Auth Key – this is the authentication secret used to request and obtain the OAuth Access Token.

  This property is required only when performing headless authentication in OAuth 1.0. It can be obtained from the **Get OAuth Authorization Url** stored procedure.

  It can be supplied alongside the **Auth Token** in the **Get OAuth Access Token** stored procedure to obtain the **OAuth Access Token**.

- OAuth Params – this is the comma-separated list of other parameters to submit in the request for the OAuth access token in the format paramname=value.

14. Connection Property: Proxy

This property has the following settings available for configuration:

- Proxy Auto Detect – this is a True/False selection. Select the appropriate value, based on the following determination:
  - When this connection property is set to True, the provider checks your system proxy settings for existing proxy server configurations (no need to manually supply proxy server details).

  Set to **False** if you want to manually configure the provider to connect to a specific proxy server.

  The default value is **True**.

  If required, more information on this setting and its properties can be provided.

- Proxy Server – this is the hostname or IP address of a proxy to route HTTP traffic through.

- Proxy Port – this is the TCP port the Proxy Server is running on.

  The default value is **80**.

- Proxy Auth Scheme – select from the dropdown list the authentication type to use to authenticate to the Proxy Server proxy. The options available are:
  - BASIC
  - DIGEST
  - NONE
  - NEGOTIATE
  - NTLM
  - PROPRIETARY

  The default value is **BASIC**.

If required, more information on this setting and its properties can be provided.

- Proxy User – this is a username to be used to authenticate to the Proxy Server proxy.

  The **Proxy User** and **Proxy Password** (see below) options are used to connect and authenticate against the HTTP proxy specified in **Proxy Server** (see above).
- Proxy Password – this is a password to use to authenticate to the Proxy Server proxy.

  This property is used to authenticate to an HTTP proxy server that supports NTLM (Windows), Kerberos, or HTTP authentication. To specify the HTTP proxy, you can set **Proxy Server** (see above) and **Proxy Port** (see above). To specify the authentication type, set **Proxy Auth Scheme** (see above).

  If you are using HTTP authentication, additionally set **Proxy User** (see above) and **Proxy Password** to HTTP proxy.

  If you are using NTLM authentication, set **Proxy User** (see above) and **Proxy Password** to your Windows password. You may also need these to complete Kerberos authentication.
  For SOCKS 5 authentication or tunneling, see **Firewall Type**.

  By default, the driver uses the system proxy. If you want to connect to another proxy, set **Proxy Auto Detect** (see above) to false.

- Proxy SSL Type – select from the dropdown list the SSL type to use when connecting to the Proxy Server proxy. The options available are:
  - AUTO - Default setting. If the URL is an HTTPS URL, the driver will use the TUNNEL option. If the URL is an HTTP URL, the component will use the NEVER option.
  - ALWAYS - The connection is always SSL enabled.
  - NEVER - The connection is not SSL enabled.
  - TUNNEL - The connection is through a tunneling proxy. The proxy server opens a connection to the remote host and traffic flows back and forth through the proxy.

  The default value is **AUTO**.

- Proxy Exceptions – this is a semi-colon – separated list of destination hostnames or IPs that are exempt from connecting through the Proxy Server.

15. Connection Property: Schema

This property has the following settings available for configuration:

- Location – specify the path to the directory that contains the schema files defining tables, views, and stored procedures.

  While there may be a default value already showing for this setting, you can click the **three dots** in the input box, which will then open a dialog allowing you to browse to the desired location.
  The folder location can be a relative path from the location of the executable. The **Location** property is only needed if you want to customize definitions (for example, change a column name, ignore a column, and so on) or extend the data model with new tables, views, or stored procedures.

- Browsable Schemas – this property restricts the schemas reported to a subset of the available schemas. For example, BrowsableSchemas = SchemaA, SchemaB, SchemaC.

  Listing the schemas from databases can be expensive. Providing a list of schemas in the connection string improves the performance.

- Tables – this property restricts the tables reported to a subset of the available tables. For example, Tables = TableA, TableB, Table C.

  Listing the tables from some databases can be expensive. Providing a list of tables in the connection string improves the performance of the driver.

  This property can also be used as an alternative to automatically listing views if you already know which ones you want to work with and there would otherwise be too many to work with.

  Specify the tables you want in a comma-separated list. Each table should be a valid SQL identifier with any special characters escaped using square brackets, double-quotes or backticks.

- Views – this restricts the views reported to a subset of the available tables. For example, Views = ViewA, ViewB, ViewC.

  Listing the views from some databases can be expensive. Providing a list of views in the connection string improves the performance of the driver.

  This property can also be used as an alternative to automatically listing views if you already know which ones you want to work with and there would otherwise be too many to work with.

Specify the views you want in a comma-separated list. Each view should be a valid SQL identifier with any special characters escaped using square brackets, double-quotes or backticks.

16. Connection Property: SSH

This property has the following settings available for configuration:

- SSH Auth Mode – select from the dropdown list the authentication method used when establishing an SSH Tunnel to the service. The options available are:
  - None - No authentication is performed. The current **SSH User** value is ignored, and the connection is logged in as anonymous.
  - Password - The driver uses the values of **SSH User** (see below) and **SSH Password** (see below) to authenticate the user.
  - Public Key - The driver uses the values of **SSH User** (see below) and **SSH Client Cert** (see below) to authenticate the user. **SSH Client Cert** (see below) must have a private key available for this authentication method to succeed.

  The default value is **Password**.

- SSH Client Cert – this is a certificate to be used for authenticating the SSH User (see below).

  SSH Client Cert must contain a valid private key in order to use public key authentication. A public key is optional, if one is not included then the driver generates it from the private key. The driver sends the public key to the server and the connection is allowed if the user has authorized the public key.

The **SSH Client Cert Type** (see below) field specifies the type of the key store specified by **SSH Client Cert**. If the store is password protected, specify the password in **SSH Client Cert Password** (see below).

Some types of key stores are containers which may include multiple keys. By default the driver will select the first key in the store, but you can specify a specific key using **SSH Client Cert Subject** (see below).

- SSH Client Cert Password – this is the password of the SSH Client Cert key, if it has one.

  This property is required for SSH tunneling when using certificate-based authentication. If the SSH certificate is in a password-protected key store, provide the password using this property to access the certificate.

- SSH Client Cert Subject – this is the subject of the SSH client certificate.

  When loading a certificate, the subject is used to locate the certificate in the store.

  If an exact match is not found, the store is searched for subjects containing the value of the property.

  If a match is still not found, the property is set to an empty string, and no certificate is selected.

  The special value "*" picks the first certificate in the certificate store.

The certificate subject is a comma separated list of distinguished name fields and values. For instance, "CN=www.server.com, OU=test, C=US, E=support@cdata.com".

The default value is *.

- SSH Client Cert Type – specify, by selecting from the dropdown list, the type of SSH Client Cert private key. The following options are available for selection:

  - USER
  - MACHINE
  - PFXFILE
  - PFXBLOB
  - JKSFILE
  - JKSBLOB
  - JKSFILE
  - PEMKEY_FILE
  - PEMKEY_BLOB
  - PPKFILE
  - PPKBLOB
  - XMLFILE
  - XMLBLOB

  The default value is **PEMKEY_FILE**.

  If required, more information on this setting and its properties can be provided.

- SSH User – this is the SSH user.

- SSH Password - this is the SSH password.

17. Connection Property: SSL

This property has the following settings available for configuration:

- SSL Client Cert – this is the TLS/SSL client certificate store for SSL Client Authentication (2-way SSL).

  The **SSL Client Cert Type** (see below) field specifies the type of the certificate store specified by **SSL Client Cert**. If the store is password protected, specify the password in **SSL Client Cert Password** (see below).

  SSL Client Cert is used in conjunction with the **SSL Client Cert Subject** (see below) field in order to specify client certificates. If **SSL Client Cert** has a value, and **SSL Client Cert Subject (see below)** is set, a search for a certificate is initiated. See **SSL Client Cert Subject** for more information.

- SSL Client Cert Type – this is the type of key store containing the TLS/SSL client certificate.

  The default value is **USER**.

  If required, more information on this setting and its properties can be provided.

- SSL Client Cert Password – this is the password for the TLS/SSL client certificate. If the certificate store is of a type that requires a password, this property is used to specify that password to open the certificate store.

- SSL Client Cert Subject – this is the subject of the TLS/SSL client certificate.

  When loading a certificate, the subject is used to locate the certificate in the store.

  If an exact match is not found, the store is searched for subjects containing the value of the property. If a match is still not found, the property is set to an empty string, and no certificate is selected.

  The special value "*" picks the first certificate in the certificate store.

  The certificate subject is a comma separated list of distinguished name fields and values. For example, "CN=www.server.com, OU=test, C=US, E=support@company.com".

  The default value is *.

- SSL Mode – select from the dropdown list the authentication mechanism to be used when connecting to the FTP or FTPS server. The options available are:
  - AUTOMATIC
  - NONE
  - IMPLICIT
  - EXPLICIT

  If **SSL Mode** is set to NONE, default plaintext authentication is used to log in to the server.

  If **SSL Mode** is set to IMPLICIT, the SSL negotiation will start immediately after the connection is established.

If **SSL Mode** is set to EXPLICIT, the driver will first connect in plaintext, and then explicitly start SSL negotiation through a protocol command such as STARTTLS.

If **SSL Mode** is set to AUTOMATIC, if the remote port is set to the standard plaintext port of the protocol (where applicable), the component will behave the same as if **SSL Mode** is set to EXPLICIT. In all other cases, SSL negotiation will be IMPLICIT.

The default value is **AUTOMATIC**.

- SSL Server Cert - this is the certificate to be accepted from the server when connecting using TLS/SSL.

  If using a TLS/SSL connection, this property can be used to specify the TLS/SSL certificate to be accepted from the server. Any other certificate that is not trusted by the machine is rejected.

  If not specified, any certificate trusted by the machine is accepted.

  Use '*' to signify to accept all certificates. Note that this is not recommended due to security concerns.

18. Connection Property: SSO

This property has the following settings available for configuration:

- SSO Login URL – this is the identity provider's login URL.
- SSO Properties – additional properties required to connect to the identity provider in a semi-colon – separated list. This is used with the **SSO Login URL**.

- SSO Exchange URL – this is the URL used for consuming the SAML response and exchanging it for service specific credentials.

F. Other questions and/or request for assistance

There may be times when you need to consult with the technical support team at Arbutus Software. If so, please send an email request to support@ArbutusSoftware.com.

For more information, please refer to the CONTACT US page on our website.