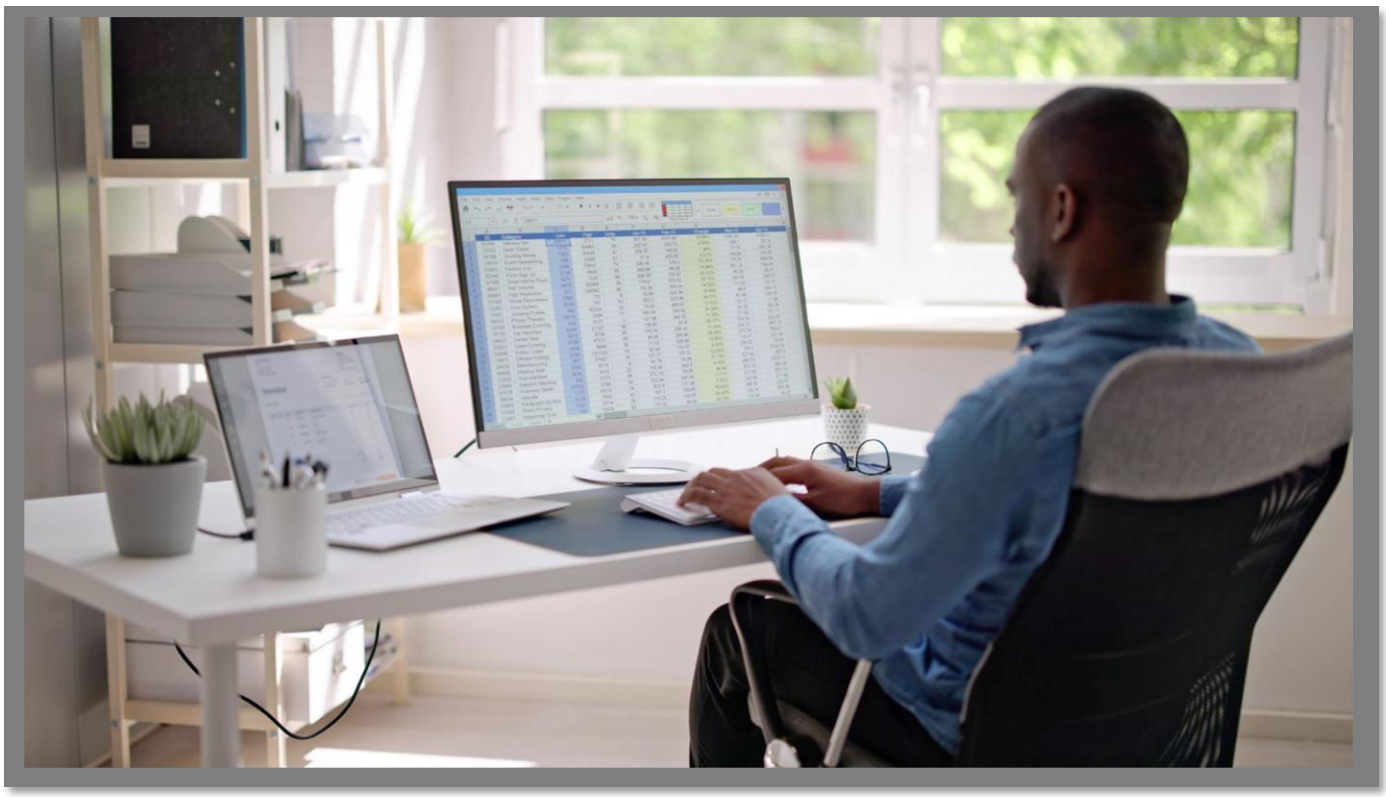
Arbutus Connectors

Salesforce
CONFIGURATION GUIDE



ARBUTUS
Powerful Analytics Simplified

Contents

A. Introduction 1

B. About Salesforce 1

C. Determining if the Connector exists prior to configuring 2

D. Configuring the Connector after it has been installed 3

E. Editing the DSN properties – the Basic and Advanced tabs. 6

E1. Editing the DSN properties in the Basic tab 6

E2. Editing the DSN properties in the Advanced tab 20

F. Other questions and/or request for assistance 22

Arbutus Connector – Salesforce

A. Introduction

The purpose of this Guide is to provide assistance with configuring the Arbutus Salesforce 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 Salesforce

Salesforce is a leading cloud-based customer relationship management (CRM) platform. It helps businesses manage their sales, marketing, customer service, and other operations by providing tools for tracking customer interactions, automating workflows, and analyzing data. Salesforce is known for its scalability, customization options, and integration capabilities with other applications.

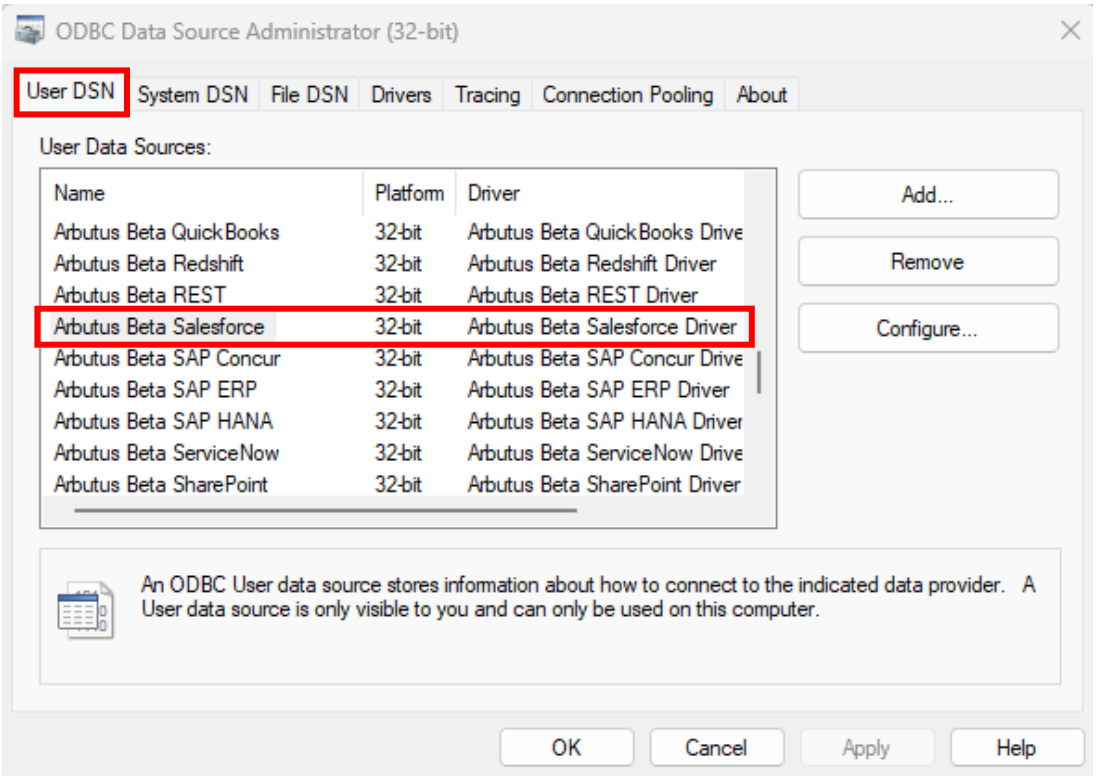
Salesforce stores data in several ways, depending on the type of data and the edition of Salesforce you are using. Salesforce uses a combination of relational databases and cloud storage solutions to manage and store data securely and efficiently.

C. Determining if the Connector exists prior to configuring

Installation of the Arbutus Salesforce 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**.



- If the Arbutus Salesforce 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.

Arbutus Connectors

- ***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.

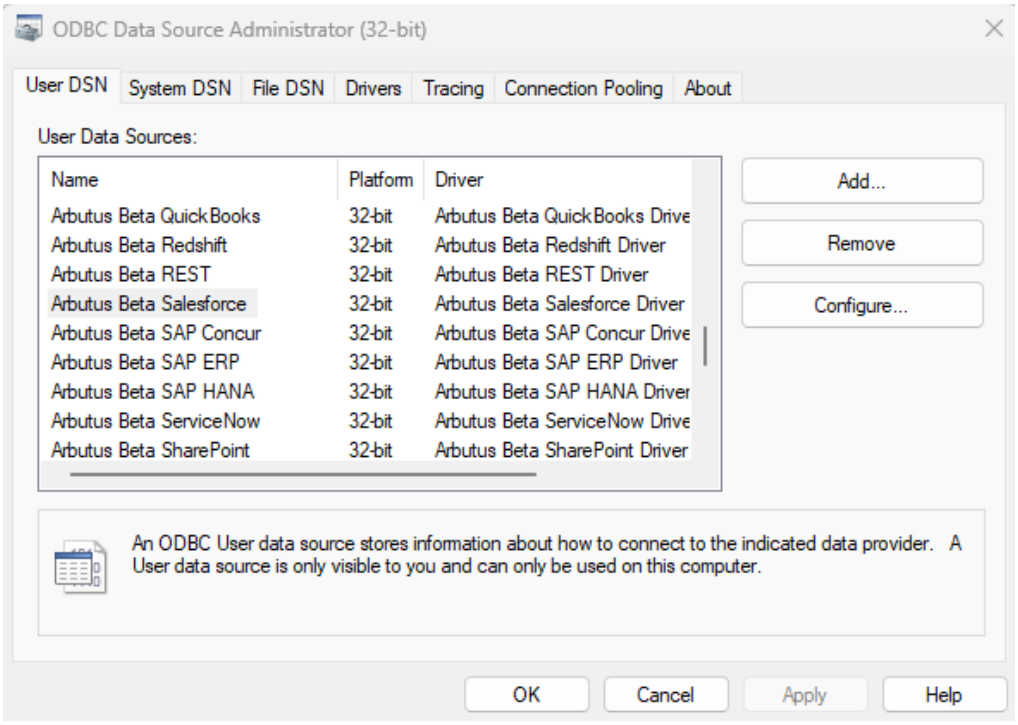
Each of these DSNs has a tab in the **ODBC Data Source Administrator** dialog.

The Arbutus ODBC Driver for Salesforce enables real-time access to Salesforce data, directly from any applications that support ODBC connectivity, the most widely supported interface for connecting applications with data.

Arbutus Connectors

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

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



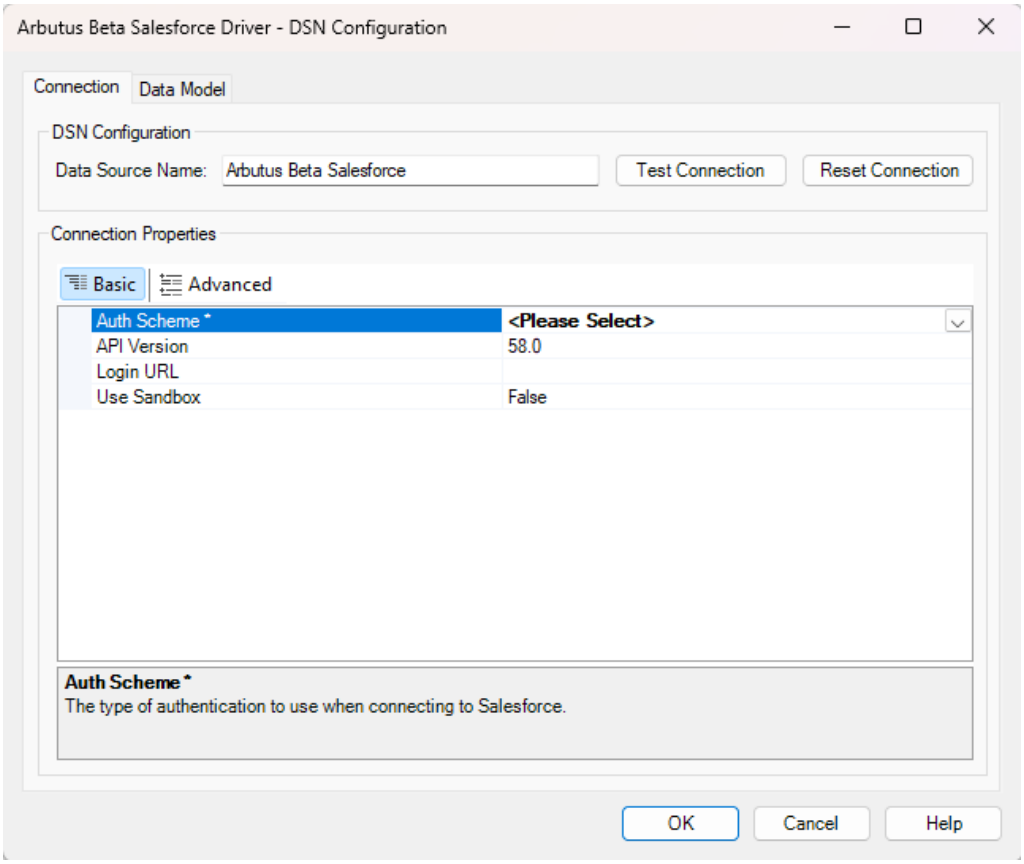
2. Click the **User DSN** tab.

   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 Salesforce.**

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

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



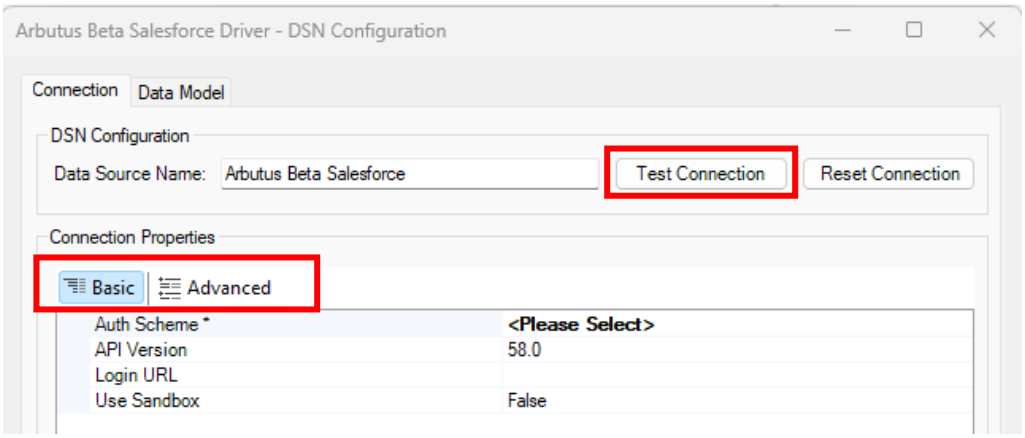
E. Editing the DSN properties – the Basic and Advanced tabs

With the DSN Configuration dialog open, the next step is to edit the DSN properties, where necessary, in the **Basic** and **Advanced** tabs. For example, editing the **Auth Scheme properties** (per screenshot below) to ensure correct authentication to the server is applied.

E1. Editing the DSN properties in the Basic tab

The properties listed in the **Basic** tab are typically the ones that are most commonly used, and as such are designed to be more user-friendly and straightforward, allowing you to quickly make changes without needing in-depth technical knowledge.

Once you have completed editing the properties in the **Basic** tab, you can go ahead and try testing the connection to the Salesforce system by clicking the **Test Connection** button, as highlighted in the screenshot below.



In the **Basic** tab, there are **four** main properties to review:

1. **Auth Scheme** – click the dropdown to select from the list the appropriate scheme used for authentication. The options available for selection are as follows:
   - Basic – select this if you want to use basic authentication. Some of the key reasons to use this option include:
     - Basic authentication is straightforward to implement, making it suitable for simple scripts, manual API calls, or quick testing.
     - It allows you to directly use your Salesforce username and password (or an API token) to authenticate, which can be convenient for quick access.
     - It is widely supported by various tools and libraries, making it easy to integrate with different systems.

     However, it is **important to note** that basic authentication is less secure compared to other methods like OAuth or API tokens, as it involves sending credentials with each request. For production environments, it is recommended to use more secure authentication methods.

Selecting **Basic** requires you to specify the following:

- User – this is the Salesforce user account used to authenticate. Together with **Password**, this field is used to authenticate against the Salesforce server

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

- Security Token – this is the security token used to authenticate access to the Salesforce account.

  Together with **User** and **Password**, this field can be used to authenticate against the Salesforce server. This is only required if your organization is setup to require it. A security token can be obtained by going to your profile information and resetting your security token. If your password is reset, you will also need to reset the security token.

- OAuth – select this if you want to use OAuth for authentication. Some of the key reasons to use this option include:
  - OAuth provides a secure way to authorize access to Salesforce without sharing your username and password. It uses tokens to grant access, which can be limited in scope and duration.
  - OAuth is ideal for integrating Salesforce with third-party applications, allowing them to access Salesforce data on behalf of users without exposing their credentials.
  - OAuth requires user consent for access, ensuring that users are aware of and approve the permissions granted to the application.
  - OAuth is a widely adopted standard for authorization, making it easier to implement and maintain across different systems and applications.

Selecting **OAuth** requires you to specify the following:

- API Version – this is the version of the Salesforce API used.

  The default value is **58.0**.

- Login URL – this is the URL to the Salesforce server used for logging in.

- Use Sandbox - this is a True/False selection. Select the appropriate value, based on following determination:
  A boolean determining if the connection should be made to a Salesforce sandbox account.

  To connect to a Salesforce sandbox account, set UseSandbox = true and append the sandbox name to the end of the username. For example, if your user name is "user", and sandbox name in Salesforce is "sandbox", the specified User should appear as "user.sandbox"

  The default value is **False**.

- OAuth Password – select this if you want to use OAuth 2.0 Username-Password flow for authentication. Some of the key reasons to use this option include:
  - This flow allows you to directly authenticate using your Salesforce username and password, which can be convenient for applications that already have the user's credentials.
  - Unlike other OAuth flows that require user interaction to grant access, the Username-Password flow can be used in scenarios where user interaction is not possible or practical.
  - This flow is useful in specific cases where other OAuth flows are not suitable, such as when the client is a trusted first-party app and there is a high degree of trust between the resource owner and the client.

However, it is **important to note** that this flow is generally less secure than other OAuth flows because it involves passing credentials back and forth. It should only be used when there is a high degree of trust and other flows are not available.

Selecting **OAuth Password** requires you to specify the following:

- User – for more information, please see this same property listed and described above (under the **OAuth** section).
- Password – for more information, please see this same property listed and described above (under the **OAuth** section).
- Security Token – for more information, please see this same property listed and described above (under the **OAuth** section).

- OAuth JWT – select this if you want to use the OAuth 2.0 JSON Web Token (JWT) Bearer flow for authentication. Some of the key reasons to use this option include:
  - The OAuth JWT Bearer flow is ideal for server-to-server integrations where you need to authorize servers to access Salesforce data without interactive user login.
  - This flow allows you to authenticate without requiring user interaction each time, making it suitable for automated processes and background services.
  - The JWT Bearer flow uses a certificate to sign the JWT request, ensuring secure transmission of identity and security information.
  - Salesforce issues an access token based on the JWT, which can be used to access protected resources. This token is short-lived, reducing the risk of credential exposure.

Selecting **OAuth JWT** requires you to specify the following:

- OAuth JWT Cert – this is 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.

  Designations of certificate stores are platform dependent.

  The following are designations of the most common User and Machine certificate stores in Windows:

▪ MY	▪ A certificate store holding personal certificates with their associated private keys
▪ CA	▪ Certifying authority certificates
▪ ROOT	▪ Root certificates
▪ SPC	▪ Software publisher certificates

  In Java, the certificate store normally is a file containing certificates and optional private keys.

  When the certificate store type is **PFXFILE**, this property must be set to the name of the file. When the type is **PFXBLOB**, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

- OAuth JWT Cert Type – click the dropdown to select from the list the type of key store containing the JWT Certificate. The different certificate store types available for selection on this dropdown are:

  USER, MACHINE, PFXFILE, PFXBLOB, JKSFILE, JKSBLOB, PEMKEY_FILE, PEMKEY_BLOB, PUBLIC_KEY_FILE, PUBLIC_KEY_BLOB, SSHPUBLIC_KEY_FILE, SSHPUBLIC_KEY_BLOB, P7BFILE, PPKFILE, XMLFILE, XMLBLOB, BCFKSFILE, BCFKSBLOB

  The default value is **PFXFILE**.

  If required, more information on each of the different certificate store types listed above can be provided.

- OAuth JWT Issuer – this is the issuer of the Java Web Token. Set this to the **OAuth Client ID**.

- OAuth JWT Cert Password – this is 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.

- 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 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@*abc_company*.com"

Note: If a field value contains a comma, it must be quoted.

If required, more information on the common fields, e.g., CN, OU, etc., and their meanings can be provided.

- OAuth JWT Subject – this is the user subject for which the application is requesting delegated access. Set this to the username (email address) of the user who is permitted to use the OAuth App.

- OAuth PKCE (Proof Key for Code Exchange) – select this if you want to enhance the security of the OAuth 2.0 authorization code flow. Some of the key reasons to use this option include:
  - PKCE adds an extra layer of security by ensuring that the client that initiates the authorization request is the same client that completes it. This is achieved by using a code verifier and code challenge.
  - PKCE helps protect against interception attacks where an attacker might try to intercept the authorization code and exchange it for an access toke.
  - PKCE is particularly useful for public clients, such as mobile apps and single-page applications, which cannot securely store a client secret.

Using **OAuth PKCE** is a best practice for securing OAuth 2.0 authorization flows, especially in scenarios where client secrets cannot be securely stored.

Selecting **OAuth PKCE** requires you to specify the following:

- 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 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.

- One Login – select this if you want to use OneLogin for Single Sign-On (SSO) authentication. Some of the key reasons to use this option include:
  - OneLogin allows users to authenticate once and gain access to multiple applications, including Salesforce, without needing to re-enter credentials.
  - OneLogin helps manage user identities and access permissions centrally, simplifying the administration of user accounts and enhancing security.
  - By integrating with OneLogin, you can leverage its advanced security features, such as multi-factor authentication (MFA) and adaptive authentication, to protect your Salesforce instance.
  - OneLogin's robust auditing and reporting features help ensure compliance with various regulatory requirements and provide visibility into user activities.

Selecting **One Login** requires you to specify the following:

- User – this is the Salesforce user account used to authenticate. Together with **Password**, this field is used to authenticate against the Salesforce server.

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

- SSO (Single Sign-On) Properties – Additional properties required to connect to the identity provider, formatted as a semicolon-separated list.

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

  The ODBC Driver for Salesforce will use the URL specified here to consume a SAML response and exchange it for service specific credentials. The retrieved credentials are the final piece during the SSO connection that are used to communicate with Salesforce.

- Ping Federate – select this when you are using **PingFederate** as your Single Sign-On (SSO) provider for Salesforce. This setup is typically chosen for enhanced security and streamlined user authentication. Some of the key reasons to use this option include:
  - If your organization uses PingFederate for SSO, selecting this property allows users to authenticate to Salesforce using their existing SSO credentials.
  - It helps in managing user access centrally, reducing the need for multiple passwords and improving security.
  - Using PingFederate can help meet compliance requirements and enhance security by leveraging robust authentication mechanisms.

Note:
PingFederate is a highly regarded enterprise federation server that specializes in user authentication and providing standardized single sign-on (SSO) solutions.

Selecting **Ping Federate** requires you to specify the following:

- User – this is the Salesforce user account used to authenticate. Together with **Password**, this field is used to authenticate against the Salesforce server.

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

- SSO (Single Sign-On) Login URL – this is the identity provider's login URL.

- SSO Properties – Additional properties required to connect to the identity provider, formatted as a semicolon-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.

  The ODBC Driver for Salesforce will use the URL specified here to consume a SAML response and exchange it for service specific credentials. The retrieved credentials are the final piece during the SSO connection that are used to communicate with Salesforce.

- OKTA – select this when you are using **Okta** as your Single Sign-On (SSO) provider for Salesforce. This setup is typically chosen for enhanced security and streamlined user authentication. Some of the key reasons to use this option include:
  - If your organization uses Okta for SSO, selecting this property allows users to authenticate to Salesforce using their existing Okta credentials.
  - It helps in managing user access centrally, reducing the need for multiple passwords and improving security.
  - Using Okta can help meet compliance requirements and enhance security by leveraging robust authentication mechanisms.

  Note:
  Okta is a widely used cloud-based identity and access management (IAM) service that provides secure identity management and Single Sign-On (SSO) solutions. It helps organizations manage user authentication and authorization across various applications and services, ensuring secure access and streamlined user experience.

  Selecting **OKTA** requires you to specify the following:
  - User – this is the Salesforce user account used to authenticate. Together with **Password**, this field is used to authenticate against the Salesforce server.
  - Password – this is the password used to authenticate the user. The **User** and **Password** are together used to authenticate with the server.
  - SSO Login URL – this is the identity provider's login URL.
  - SSO Properties - additional properties required to connect to the identity provider in a semicolon-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.

  The ODBC Driver for Salesforce will use the URL specified here to consume a SAML response and exchange it for service specific credentials. The retrieved credentials are the final piece during the SSO connection that are used to communicate with Salesforce.

- ADFS (Active Directory Federation Services) – select this when you are using **ADFS** as your Single Sign-On (SSO) provider for Salesforce. This setup is typically chosen for enhanced security and streamlined user authentication. Some of the key reasons to use this option include:
  - If your organization uses ADFS for SSO, selecting this property allows users to authenticate to Salesforce using their existing ADFS credentials.
  - It helps in managing user access centrally, reducing the need for multiple passwords and improving security.
  - Using ADFS can help meet compliance requirements and enhance security by leveraging robust authentication mechanisms.

  Selecting **ADFS** requires you to specify the following:
  - User – for more information, please see this same property listed and described above (under the **OKTA** section)..
  - Password – for more information, please see this same property listed and described above (under the **OKTA** section).
  - SSO Login URL – for more information, please see this same property listed and described above (under the **OKTA** section).

- SSO Properties - for more information, please see this same property listed and described above (under the **OKTA** section).

- SSO Exchange URL - for more information, please see this same property listed and described above (under the **OKTA** section).

- AzureAD – select this when you are using **Azure Active Directory (Azure AD)** as your Single Sign-On (SSO) provider for Salesforce. This setup is typically chosen for enhanced security and streamlined user authentication. Some of the key reasons to use this option include:
  - If your organization uses Azure AD for SSO, selecting this property allows users to authenticate to Salesforce using their existing Azure AD credentials.
  - It helps in managing user access centrally, reducing the need for multiple passwords and improving security.
  - Using Azure AD can help meet compliance requirements and enhance security by leveraging robust authentication mechanisms.

  Selecting **AzureAD** requires you to specify the following:
  - SSO Properties - for more information, please see this same property listed and described above (under the **OKTA** section).

  - SSO Exchange URL – for more information, please see this same property listed and described above (under the **OKTA** section).

  - OAuth Client ID - for more information, please see this same property listed and described above (under the **OAuth PKCE** section).

- OAuth Client Secret - for more information, please see this same property listed and described above (under the **OAuth PKCE** section).

2. **API Version** – this is the version of the Salesforce API used.

   The default value is **58.0**.

3. **Login URL** – this is the URL to the Salesforce server used for logging in.

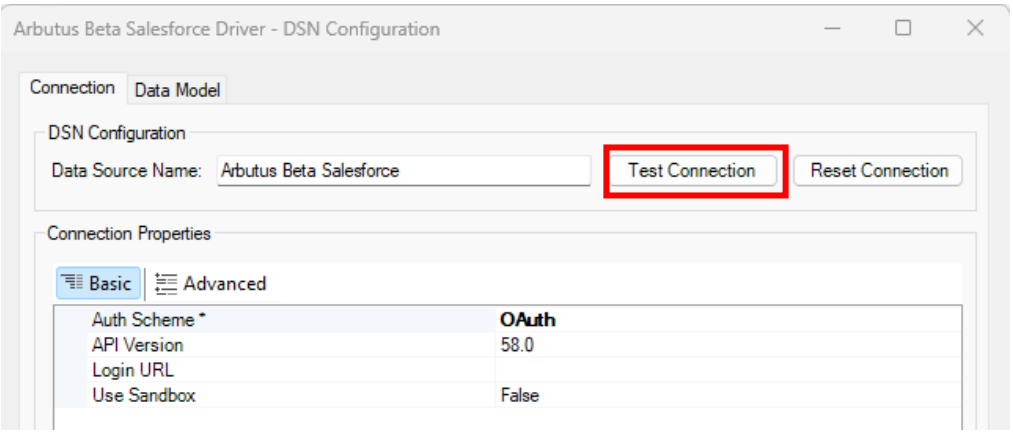
4. **Use Sandbox** - this is a True/False selection. Select the appropriate value, based on following determination:
   - A boolean determining if the connection should be made to a Salesforce sandbox account.

   To connect to a Salesforce sandbox account, set UseSandbox = true and append the sandbox name to the end of the username. For example, if your user name is "user", and sandbox name in Salesforce is "sandbox", the specified User should appear as "user.sandbox"

E2. Editing the DSN properties in the Advanced tab

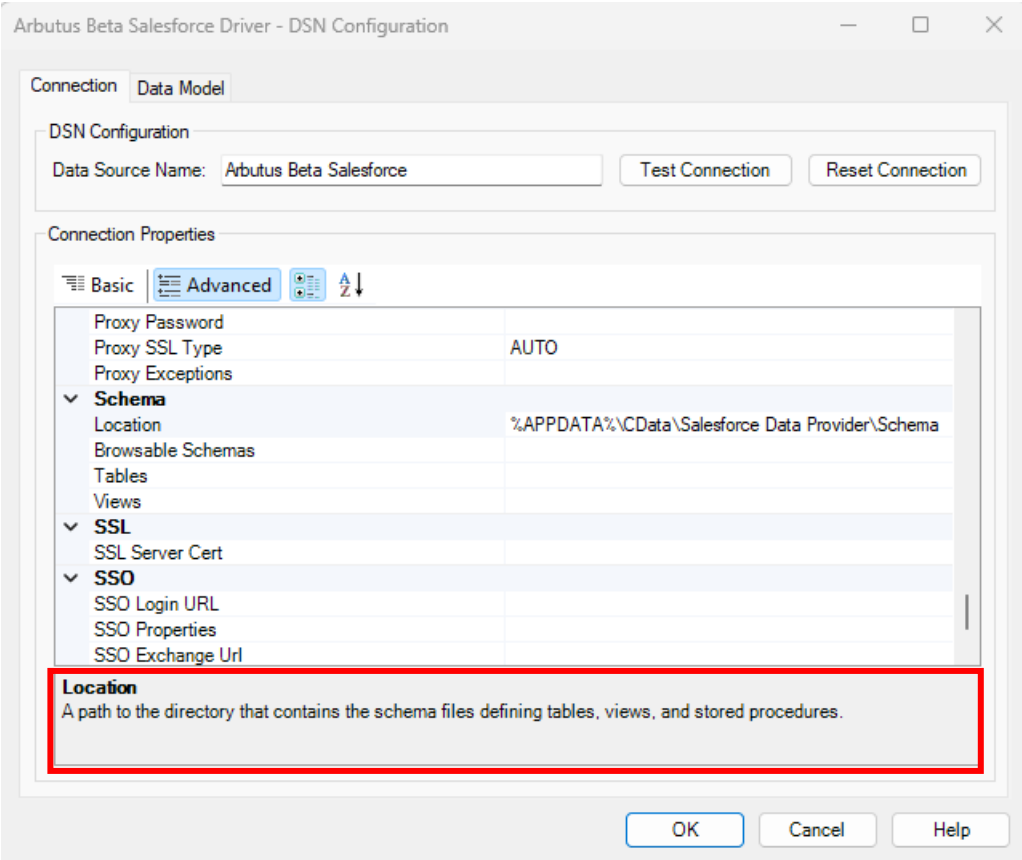
This tab includes more detailed and technical properties. It is intended for those users who need more control over the configuration and are comfortable with more complex options. The **Advanced** tab often includes properties that can fine-tune the behaviour of the system feature.

If you have already completed editing the properties in the **Basic** tab, as required, you do not necessarily need to also complete editing the properties in the **Advanced** tab. Instead, once you have completed editing the properties in the **Basic** tab, you may opt to proceed to testing the connection to the Salesforce system by clicking the **Test Connection** button.



There are a lot more properties included for editing in the **Advanced** tab.

However, it is useful to know that each property does provide a short description of it and as such serves as a guide in terms of what to edit and/or enter. This short description can be seen at the bottom of the **DSN Configuration** dialog box, as seen in the screenshot below.



If it is deemed necessary to complete some/all the properties in the **Advanced** tab, it is recommended that you refer to the description shown for any of the properties being edited and/or entered.

If required, more information on the properties listed in the **Advanced** tab can also be provided.

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.