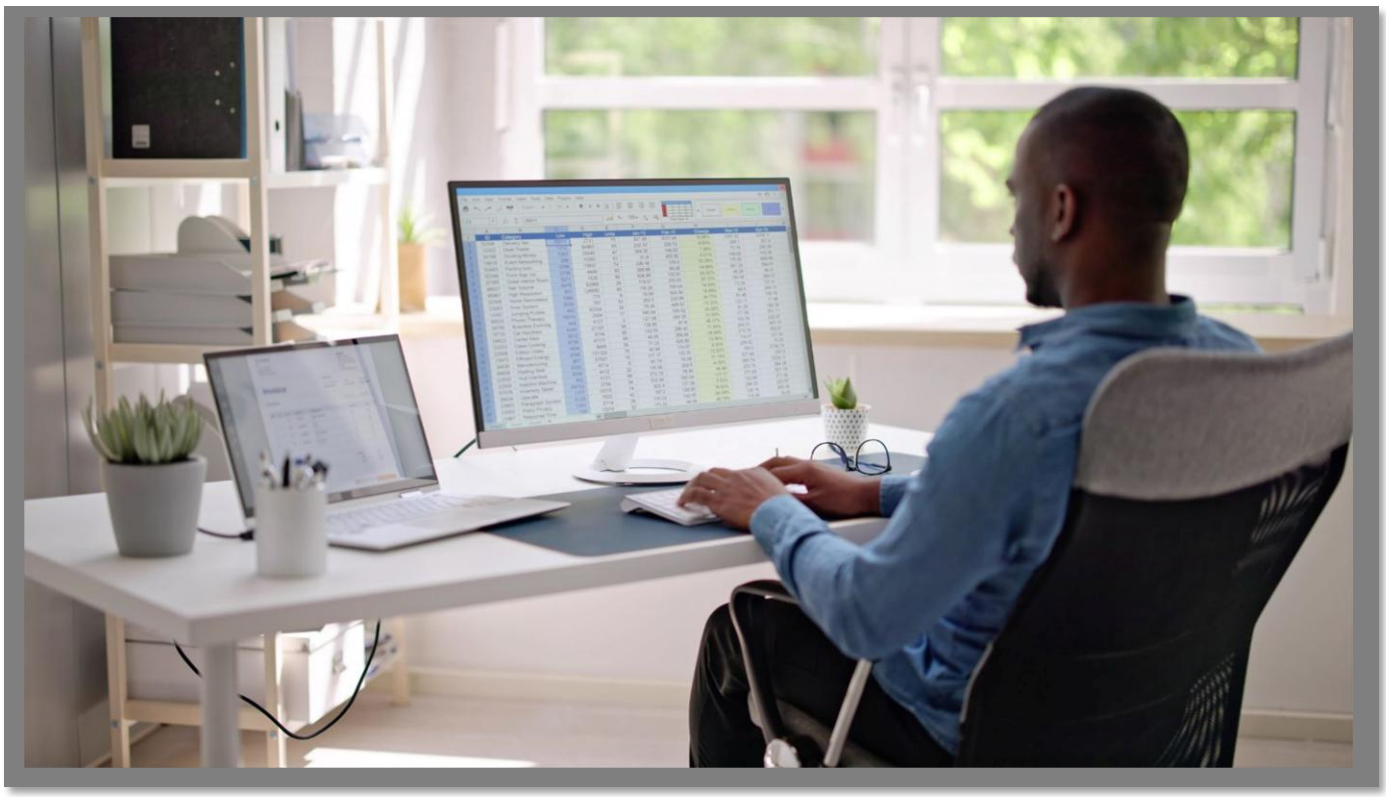
Arbutus Connectors

PostgreSQL
CONFIGURATION GUIDE



ARBUTUS
Powerful Analytics Simplified

Contents

Arbutus Connector – PostgreSQL

A. Introduction

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

PostgreSQL is a powerful, open-source relational database management system (RDBMS). It is known for its robustness, extensibility, and standards compliance. PostgreSQL supports a wide range of data types and advanced features like complex queries, foreign keys, triggers, and updatable views. It is widely used for both small and large-scale applications due to its reliability and performance.

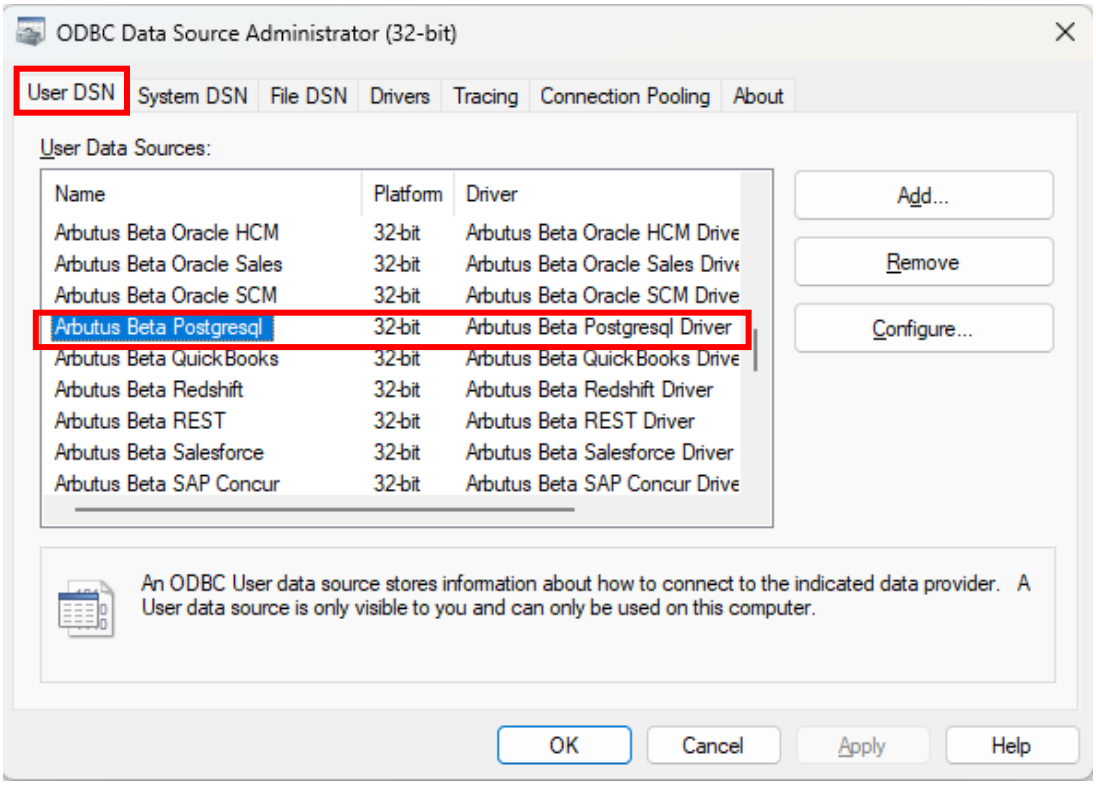
In PostgreSQL, data is stored in a structured format within a database. Data is organized into tables and tables are grouped into schemas, which help organize and manage the database objects.

C. Determining if the Connector exists prior to configuring

Installation of the Arbutus PostgreSQL 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 PostgreSQL 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.

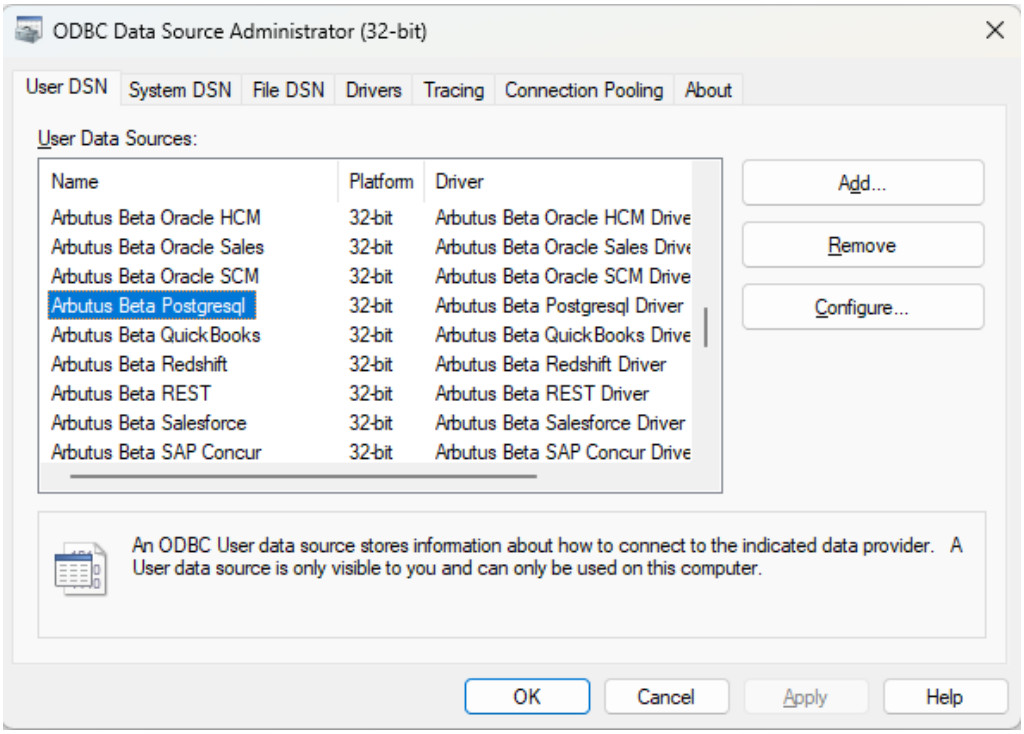
Each of these DSNs has a tab in the **ODBC Data Source Administrator** dialog.

The ODBC Driver for PostgreSQL enables real-time access to PostgreSQL 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**.



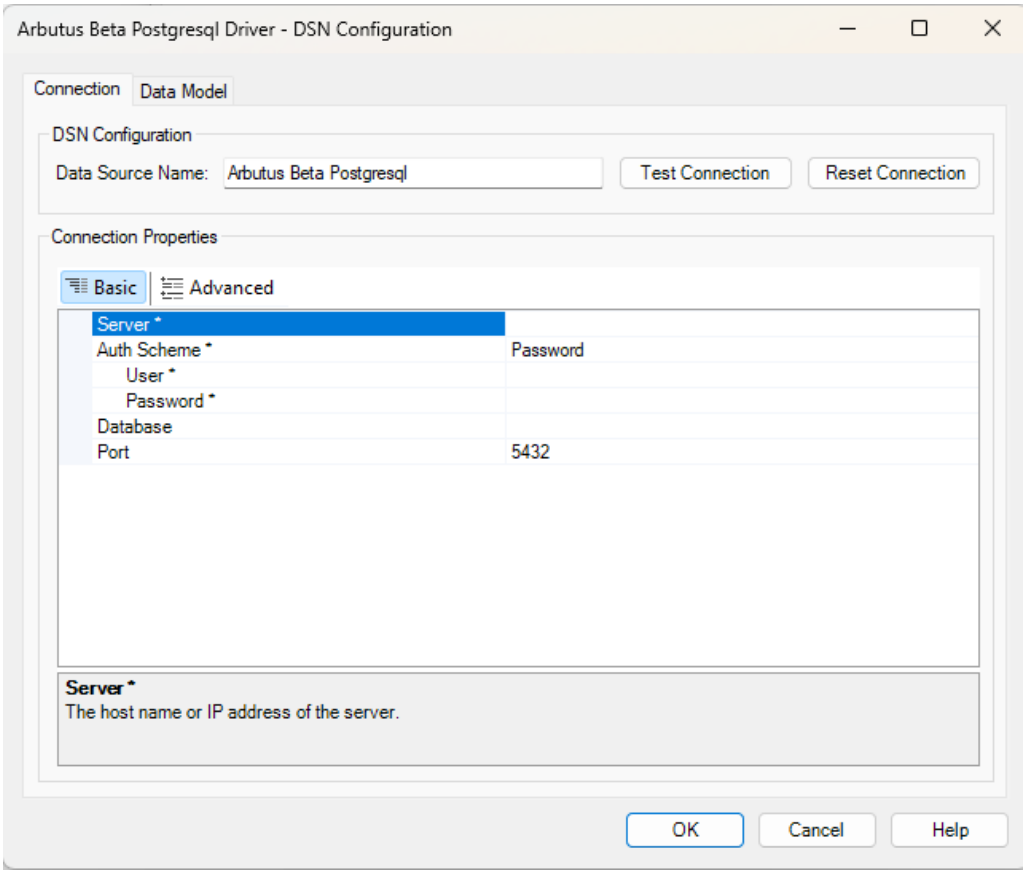
2. Click the **User DSN** tab.

   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 PostgreSQL.**

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

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



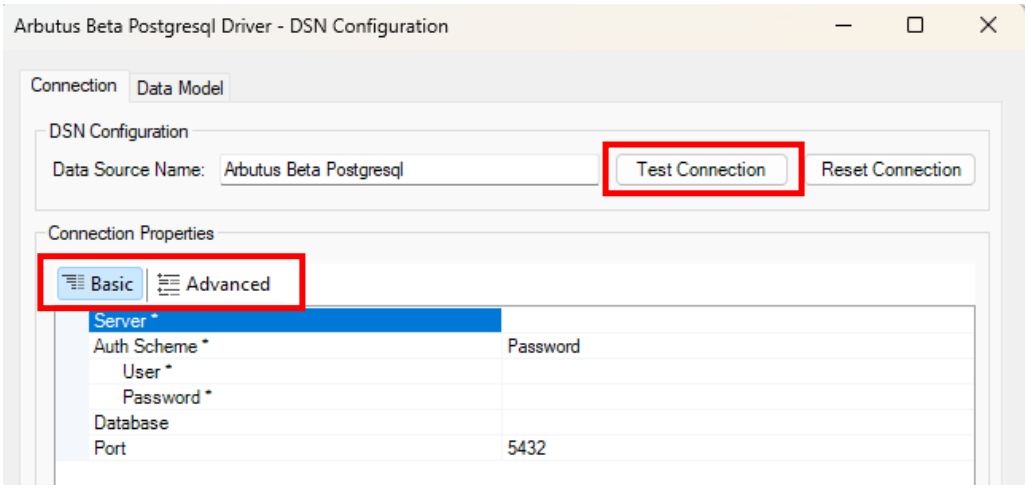
E. Editing the DSN properties – the Basic and Advanced tabs

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 **5432** entry for the **Port** (per screenshot below) to match the Port of the Client's PostgreSQL Server.

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 PostgreSQL Server by clicking the **Test Connection** button, as highlighted in the screenshot below.



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

1. **Server** – enter the host name or IP address of the server. If not set, the default value "localhost" is used.

2. **Auth Scheme** – click the dropdown to select from the list the appropriate scheme used for authentication. The options available for selection are as follows:

    - Password – select this when you want to authenticate using a **username and password** combination. This is a common method for connecting to PostgreSQL databases, especially when you have a specific user account set up with the necessary permissions to access the database.

      This method is straightforward and widely supported, making it suitable for many scenarios, including local development and production environments where you manage user credentials directly.

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

- User – this is the PostgreSQL user account used to authenticate. The authenticating server requires both **User** and **Password** (see below) to validate the user's identity.

- Password - this is the password used to authenticate the user. The authenticating server requires both **User** (see above) and **Password** to validate the user's identity.

The default value is **Password**.

- AzureAD - select this when you are using **Azure Active Directory (AAD)** for authentication. This is particularly useful if your PostgreSQL database is hosted on **Azure** and you want to leverage the security and management features provided by Azure AD.

  Using Azure AD allows you to manage user access and permissions centrally, providing a more secure and streamlined authentication process.

  Selecting AzureAD requires you to specify the following:

  - User – this is the PostgreSQL user account used to authenticate.

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

    For example, contoso.onmicrosoft.com. Alternatively, specify the tenant Id.

A tenant is a digital representation of your organization, primarily associated with a domain, for example, Microsoft.com. The tenant is managed through a Tenant ID (also known as the directory ID), which is specified whenever you assign users permissions to access or manage Azure resources.

To locate the directory ID in the Azure Portal, navigate to **Azure Active Directory > Properties**.

Specifying **AzureTenant** is required when **AuthScheme** = either **AzureServicePrincipal** or **AzureServicePrincipalCert**, or if **AuthScheme** = **AzureAD** and the user belongs to more than one tenant.

- OAuth Client Id – this is the client Id assigned when you register your application with an OAuth authorization server. This is also known as the consumer key. This ID registers the custom application with the OAuth authorization server.

  OAuth Client Id is one of a handful of connection parameters that need to be set before users can authenticate via OAuth.

- OAuth Client Secret – this is the client secret assigned when you register your application with an OAuth authorization server. This is also known as the consumer secret. This secret registers the custom application with the OAuth authorization server.

  OAuth Client Secret is one of a handful of connection parameters that need to be set before users can authenticate via OAuth.

- Callback URL – this is the OAuth callback URL users return to after authenticating to PostgreSQL via OAuth. 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.

- AzurePassword – select this when you want to authenticate using **Azure Active Directory (AAD) credentials** with a username and password. This method is useful if your PostgreSQL database is hosted on **Azure** and you prefer to manage user access through Azure AD, but without using OAuth tokens.

  This approach allows you to leverage Azure AD's security features while using a familiar username and password combination for authentication.

  Selecting **AzurePassword** requires you to specify the following:
  - User – this is the PostgreSQL user account used to authenticate. The authenticating server requires both **User** and **Password** (see below) to validate the user's identity.

  - Password - this is the password used to authenticate the user. The authenticating server requires both **User** (see above) and **Password** to validate the user's identity.

  - Azure Tenant – for more information on this property, please see **Azure Tenant** in the section above on **AzureAD**.

- AzureMSI – select this when you want to use **Azure Managed Service Identity (MSI)** for authentication. This is particularly useful if your PostgreSQL database is hosted on **Azure** and you want to leverage the security and management features provided by Azure MSI.

Using Azure MSI allows your application to authenticate to Azure services without needing to manage credentials directly, enhancing security and simplifying management.

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

- User – this is the PostgreSQL user account used to authenticate.

- AwsIAMRoles – select this when you want to use **Amazon Web Services (AWS) Identity and Access Management (IAM) Roles** for authentication. This is particularly useful if your PostgreSQL database is hosted on **Amazon RDS** and you want to leverage the security and management features provided by AWS IAM.

  Using IAM Roles allows you to manage access permissions centrally and securely without embedding credentials in your application.

  Selecting AwsIAMRoles requires you to specify the following:

  - User – this is the PostgreSQL user account used to authenticate.

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

    To access this value:

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

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

  To access this value, please refer to the five steps listed above.

- 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 Arbutus ODBC Driver for PostgreSQL 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 (Identity Access Management) user.

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

- AwsEC2Roles – select this when you want to use **AWS IAM Roles** assigned to the **EC2 instance** where your application is running. This is particularly useful if your PostgreSQL database is hosted on **Amazon RDS** and you want to leverage the security and management features provided by AWS IAM (Identity Access Management).

  Using IAM Roles assigned to the EC2 instance allows your application to authenticate to AWS services without needing to manage credentials directly, enhancing security and simplifying management.

- GCPServiceAccount – select this when you are using a **Google Cloud SQL instance** for your PostgreSQL database. This property allows you to authenticate using a Google Service Account, which is particularly useful if your PostgreSQL instance is hosted on Google Cloud Platform (GCP) and you want to leverage the security and management features provided by GCP.

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

  - 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 property can be provided.

  - OAuth JWT Cert Type – this is the type of key containing the JWT Certificate. This is a dropdown selection consisting of following the possible options you can choose from:

    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, GOOGLEJSON, GOOGLEJSONBLOB

The default value is **USER**.

If required, more information on this property can be provided.

- OAuth JWT Cert Password – this is the password for the OAuth JWT certificate used to access a certificate store that requires a password. If the certificate store does not require a password, leave this property blank.

  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 used to locate a matching certificate in the store. Supports partial matches and the wildcard '*' to select the first certificate.

  The value of this property is used to locate a matching certificate in the store. The search process works as follows:
  - If an exact match for the subject is found, the corresponding certificate is selected.
  - If no exact match is found, the store is searched for certificates whose subjects contain the property value.
  - If no match is found, no certificate is selected.

  You can set the value to '*' to automatically select the first certificate in the 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@cdata.com.

  If required, more information on this property can be provided.

3. **Database** – enter the name of the PostgreSQL database to connect to when connecting to the PostgreSQL server. If a database is not provided, the user's default database will be used.

   If this is not set, tables from all databases will be returned.

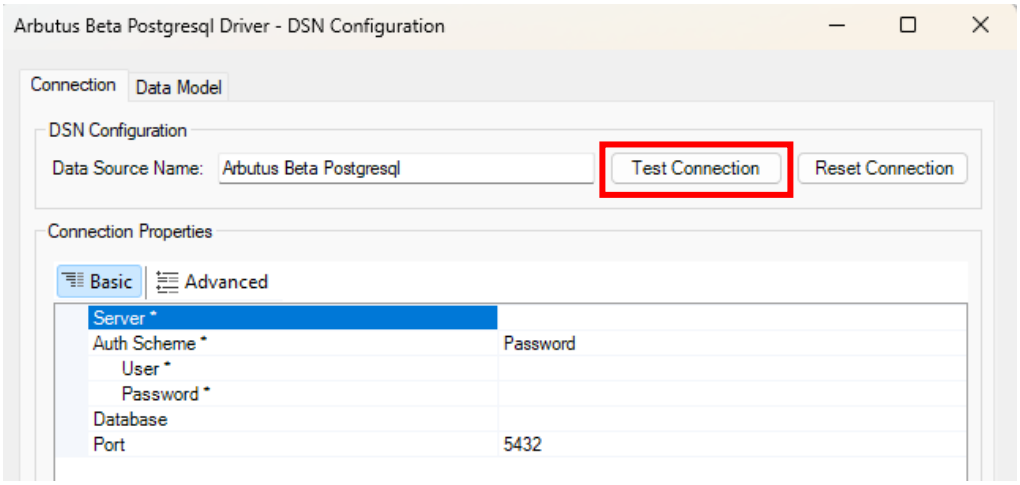
4. **Port** – enter the port of the PostgreSQL Server.

   The default value is **5432**.

E2. Editing the DSN properties in the Advanced tab

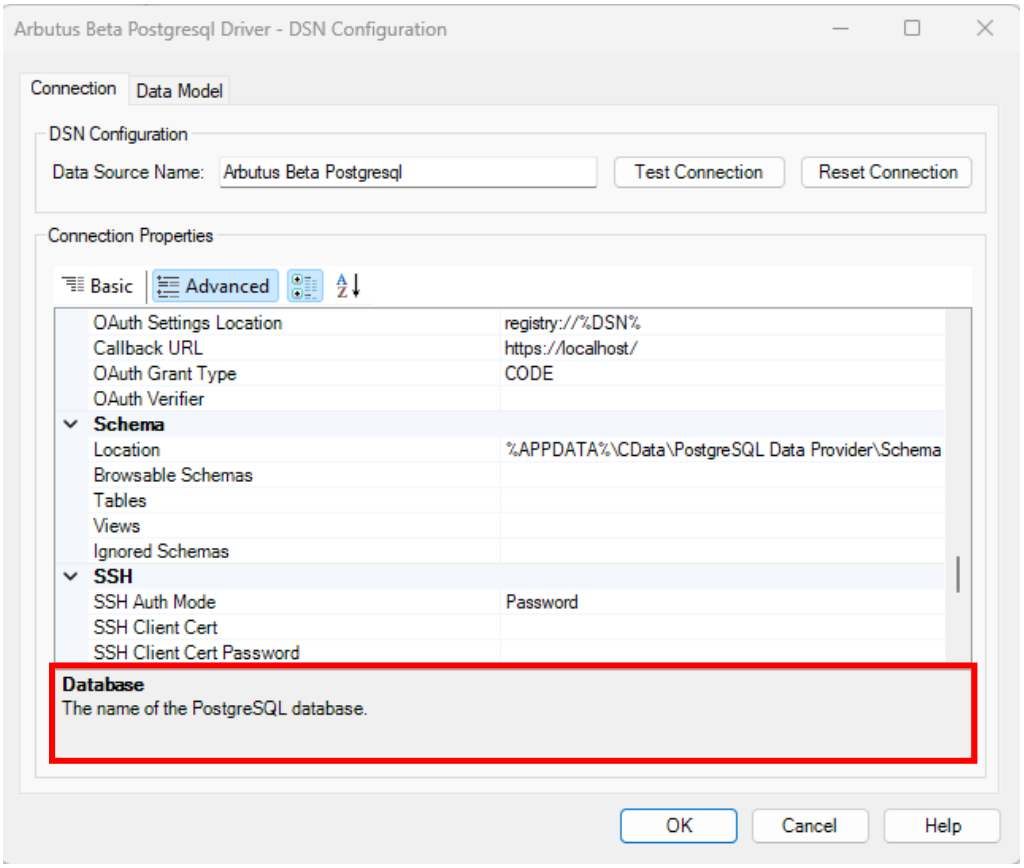
This tab includes more detailed and technical properties. It is intended for those users who need more control over the configuration and are comfortable with more complex options. The **Advanced** tab often includes properties that can fine-tune the behaviour of the system feature.

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 PostgreSQL Server 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.