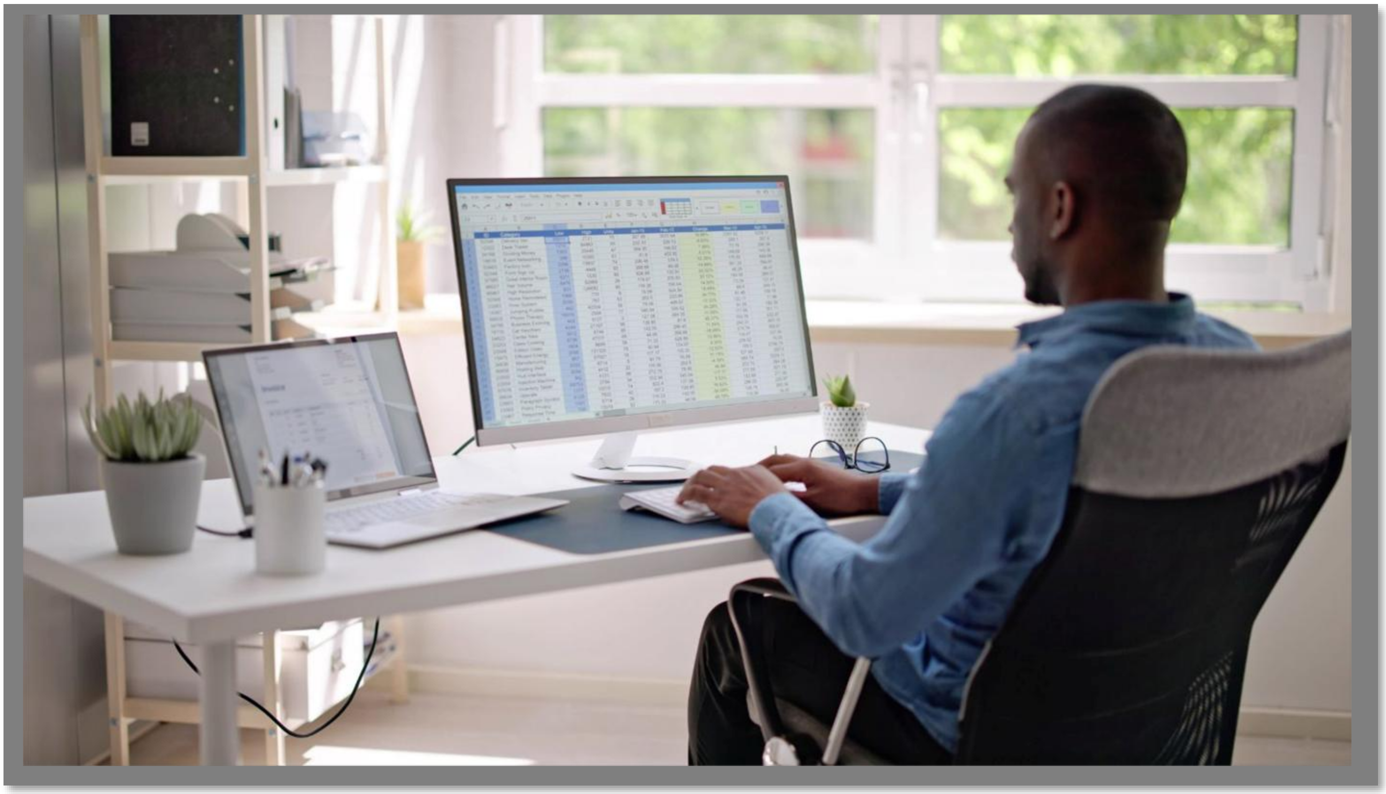
Arbutus Connectors

MySQL

CONFIGURATION GUIDE



ARBUTUS

Powerful Analytics Simplified

Arbutus Connectors

Contents

Arbutus Connector – MySQL

A. Introduction

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

MySQL is an open-source relational database management system (RDBMS). It uses Structured Query Language (SQL) to manage and manipulate data. MySQL is known for its reliability, performance, and ease of use, making it a popular choice for web applications and various software projects.

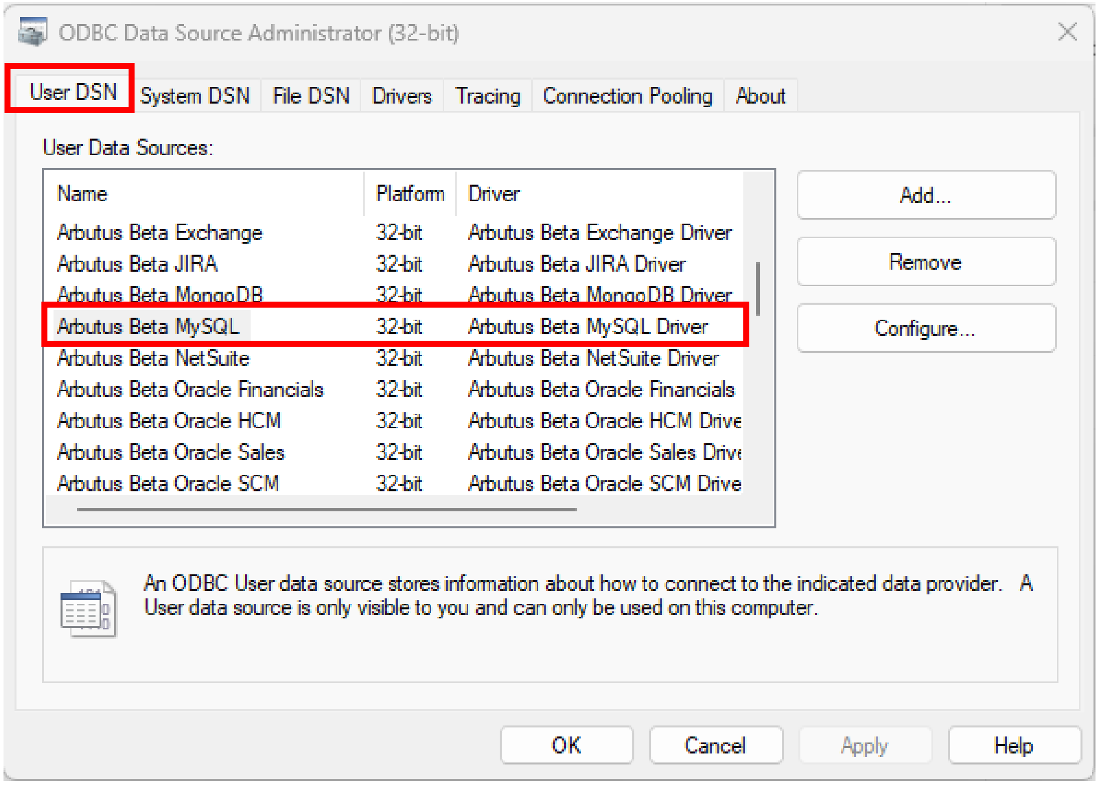
As a comparison, SQL Server, developed by Microsoft, is a commercial product with robust security features, advanced backup tools, and strong integration with Windows environments. While MySQL is often preferred for web applications, SQL Server is favored for large enterprise solutions. However, both have extensive community and professional support.
MySQL stores data in tables, organized into databases. Each table consist of rows, columns, and indexes.

C. Determining if the Connector exists prior to configuring

Installation of the Arbutus MySQL 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**.



Name	Platform	Driver
Arbutus Beta Exchange	32-bit	Arbutus Beta Exchange Driver
Arbutus Beta JIRA	32-bit	Arbutus Beta JIRA Driver
Arbutus Beta MongoDB	32-bit	Arbutus Beta MongoDB Driver
Arbutus Beta MySQL	32-bit	Arbutus Beta MySQL Driver
Arbutus Beta NetSuite	32-bit	Arbutus Beta NetSuite Driver
Arbutus Beta Oracle Financials	32-bit	Arbutus Beta Oracle Financials
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

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 MySQL 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.

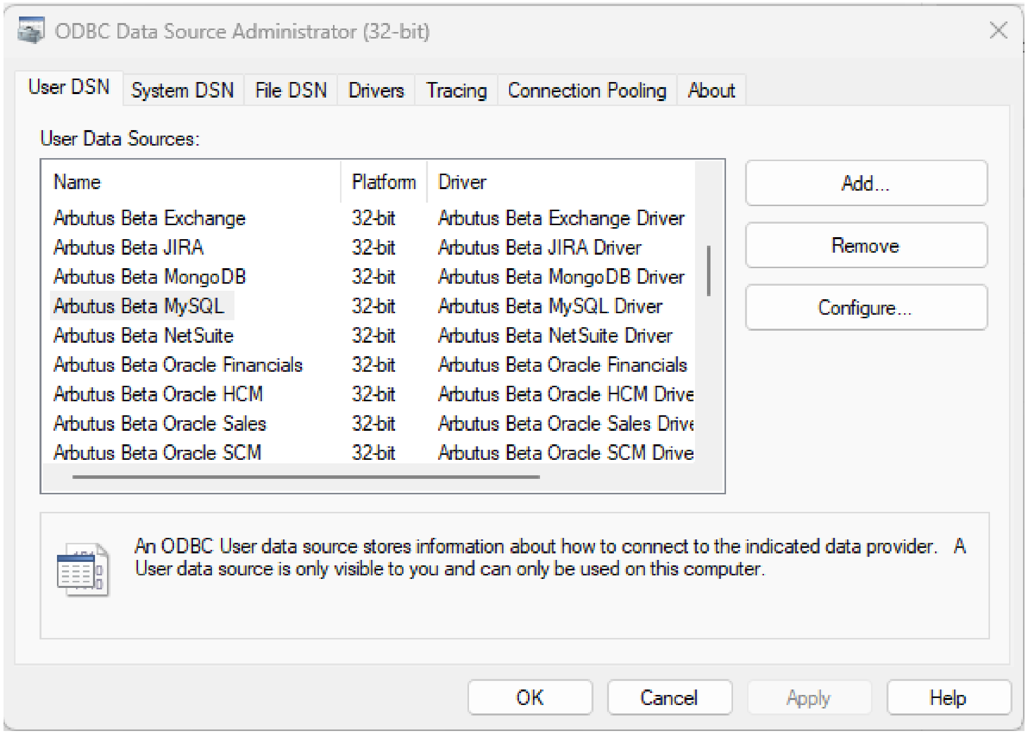
Each of these DSNs has a tab in the **ODBC Data Source Administrator** dialog.

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



Name	Platform	Driver
Arbutus Beta Exchange	32-bit	Arbutus Beta Exchange Driver
Arbutus Beta JIRA	32-bit	Arbutus Beta JIRA Driver
Arbutus Beta MongoDB	32-bit	Arbutus Beta MongoDB Driver
Arbutus Beta MySQL	32-bit	Arbutus Beta MySQL Driver
Arbutus Beta NetSuite	32-bit	Arbutus Beta NetSuite Driver
Arbutus Beta Oracle Financials	32-bit	Arbutus Beta Oracle Financials
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

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.

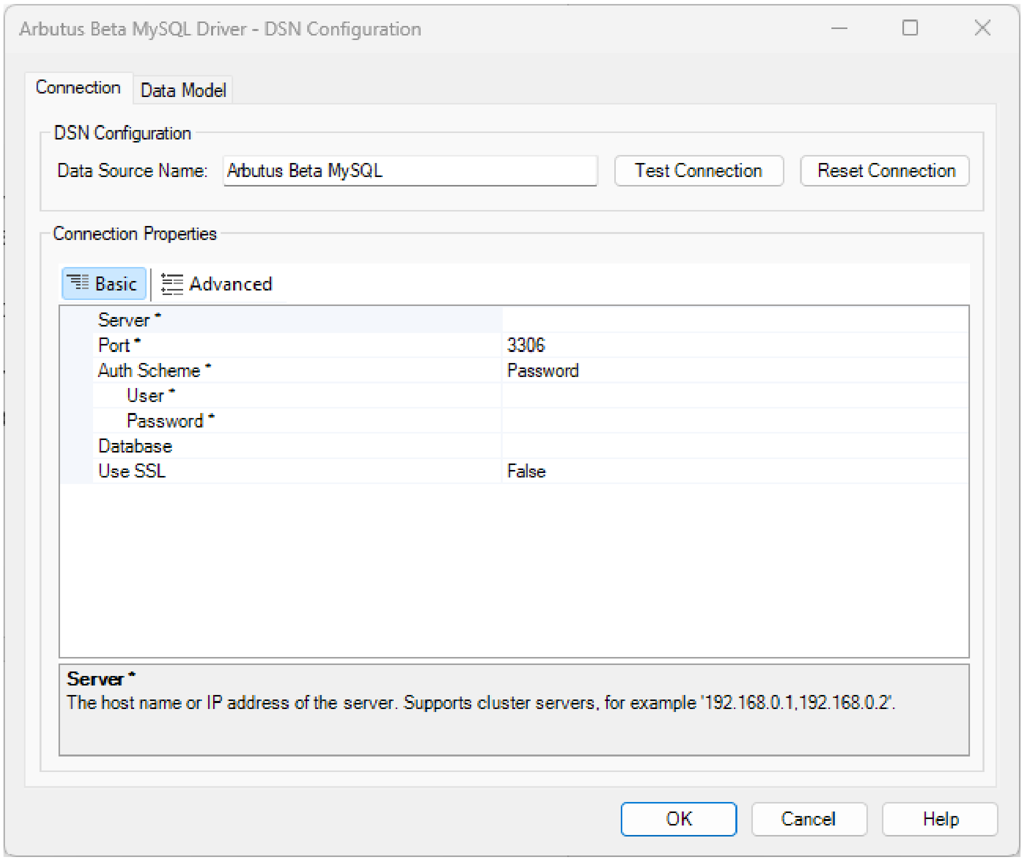
2. Click the **User DSN** tab.

   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 MySQL.**

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

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



Data Source Name: Arbutus Beta MySQL

Server *	
Port *	3306
Auth Scheme *	Password
User *	
Password *	
Database	
Use SSL	False

Server *
The host name or IP address of the server. Supports cluster servers, for example '192.168.0.1,192.168.0.2'.

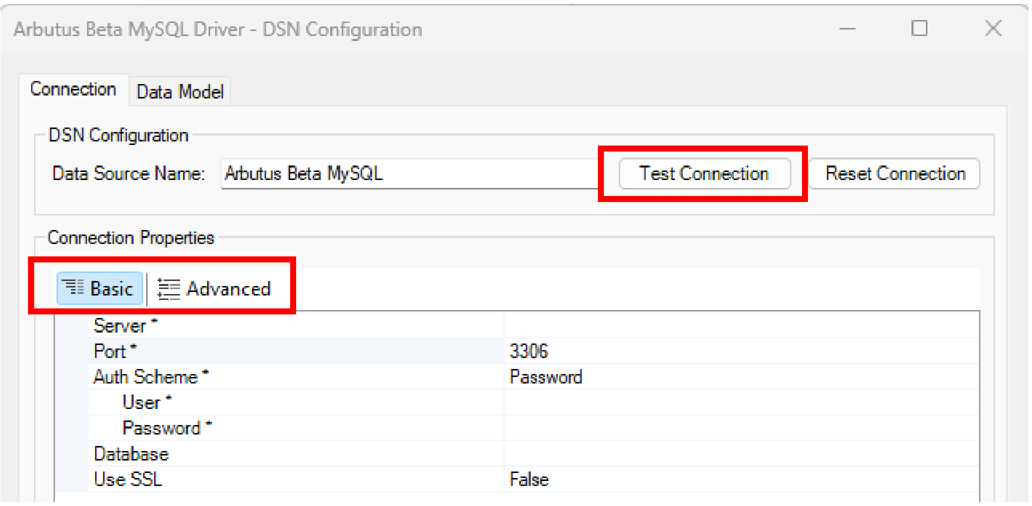
E. Editing the DSN properties – the Basic and Advanced tabs

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 **3306** entry for the **Port** (per screenshot below) to match the Port of the Client's MySQL 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 MySQL Server by clicking the **Test Connection** button, as highlighted in the screenshot below.



Data Source Name: Arbutus Beta MySQL

Server *	
Port *	3306
Auth Scheme *	Password
User *	
Password *	
Database	
Use SSL	False

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

1. **Server** – enter the host name or IP address of the server. Also supports cluster servers, for example ‘192.168.0.1, 192.168.0.2’.

2. **Port** – enter the port of the MySQL Server. Also supports cluster servers, for example ‘3306, 3307’, the number of the port should match with Server’s.

   The default value is **3306**.

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

    - Password – select this if you are using **standard username and password authentication**. This is the most common and straightforward method of authentication for MySQL databases.

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

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

      The default value is **Password**.

    - NTLM (NT Lan Manager) – select this if you are using **NT LAN Manager (NTLM)** for authentication. NTLM is a suite of Microsoft security protocols intended to provide authentication, integrity, and confidentiality to users.

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

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

  - Domain – this is the name of the domain for a Windows (NTLM) security login. By default, the driver uses the domain of the PC it is running on or the domain used by the machine running the MySQL instance.

  - NTLM Version – this property specifies the NTLM version to use. The possible values are 1 or 2.

    The default value is **1**.

- AzureAD - select this if you are using **Azure Active Directory (Azure AD)** for authentication. This is particularly relevant when your MySQL database is hosted on Azure, and you want to leverage Azure AD's authentication mechanisms.

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

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

  - 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.

    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** 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.

    - 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** (see below)

    - 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** (see above), 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.

  - AzurePassword – select this if you are using **Azure Active Directory (Azure AD) password authentication**. This method is used when you want to authenticate to your MySQL database using an Azure AD user account and password.

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

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

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

- 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.

  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** 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.

○ AzureMSI – select this if you are using **Azure Managed Service Identity (MSI)** for authentication. This method allows your application to authenticate to the MySQL database using the managed identity assigned to an Azure resource, such as a virtual machine or an Azure App Service.

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

- 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.

  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** 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.

- AwsIAMRoles – select this if you are using **AWS Identity and Access Management (IAM) roles** for authentication. This method allows your application to authenticate to the MySQL database using IAM roles assigned to AWS resources, such as EC2 instances or Lambda functions.

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

  - 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 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.

  To access this value, please refer to the 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 MySQL 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.

○ AwsEC2Roles – select this if you are using **AWS EC2 instance roles** for authentication. This method allows your application to authenticate to the MySQL database using the IAM role assigned to an EC2 instance.

- LDAP (Lightweight Directory Access Protocol) – set to LDAP to authenticate as an LDAP user. This method allows you to authenticate to your MySQL database using credentials stored in an LDAP directory, such as Microsoft Active Directory or OpenLDAP.

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

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

4. **Database** – enter the name of the default MySQL database to connect to when connecting to the MySQL Server.

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

5. **Use SSL** – this is a True/False selection. Select the appropriate value, based on following determination:
   - This field sets whether SSL is enabled.

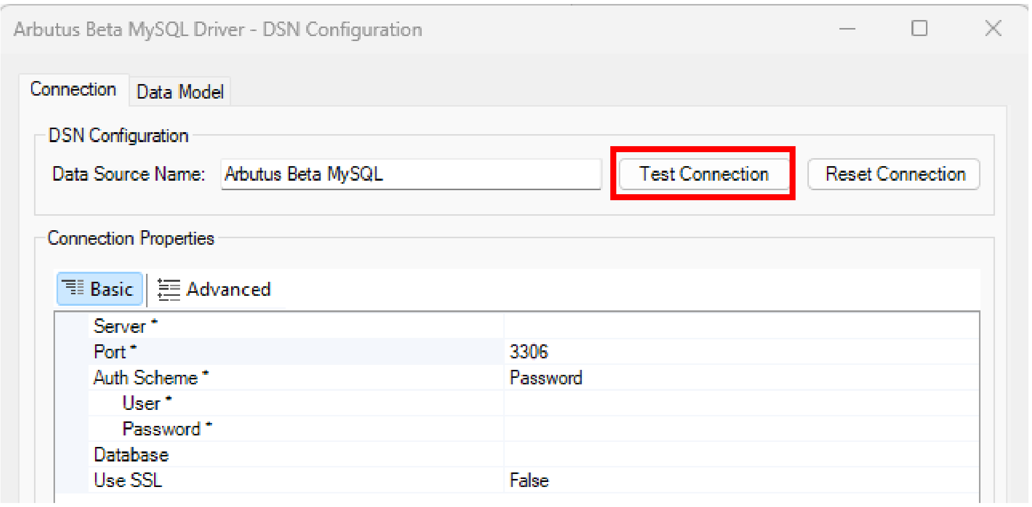
   This field sets whether the driver will attempt to negotiate TLS/SSL connections to the server. By default, the driver checks the server's certificate against the system's trusted certificate store. To specify another certificate, set **SSL Server Cert**.

   The default value is **False**.

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 MySQL Server by clicking the **Test Connection** button.

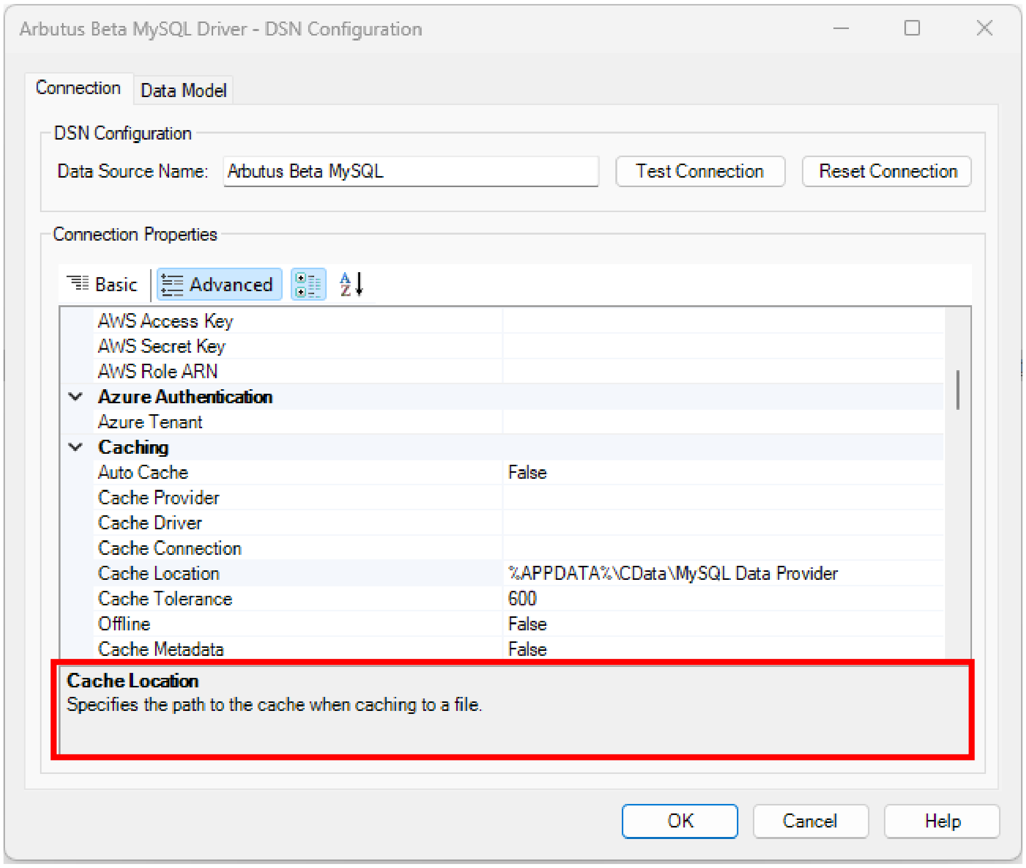


Data Source Name: Arbutus Beta MySQL

Server *	
Port *	3306
Auth Scheme *	Password
User *	
Password *	
Database	
Use SSL	False

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.



Data Source Name: Arbutus Beta MySQL

AWS Access Key	
AWS Secret Key	
AWS Role ARN	
Azure Authentication	
Azure Tenant	
Caching	
Auto Cache	False
Cache Provider	
Cache Driver	
Cache Connection	
Cache Location	%APPDATA%\CData\MySQL Data Provider
Cache Tolerance	600
Offline	False
Cache Metadata	False

Cache Location
Specifies the path to the cache when caching to a file.

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.