



Progress DataDirect for ODBC for Microsoft SharePoint User's Guide

Release 8.0.0

Copyright

Visit the following page online to see Progress Software Corporation's current Product Documentation Copyright Notice/Trademark Legend: <https://www.progress.com/legal/documentation-copyright>.

Updated: 2025/07/31

Table of Contents

Welcome to the Progress DataDirect for ODBC for Microsoft SharePoint

Driver	9
What's new in this release?.....	10
Driver requirements.....	11
Installing and setting up the driver (Windows).....	12
Installing and setting up the driver (Linux).....	15
Connection string examples.....	16
OAuth 2.0 access token flow.....	17
OAuth 2.0 refresh token grant.....	17
OAuth 2.0 dynamic authorization code grant.....	19
Proxy server.....	20
Data types.....	21
Driver specifications	22
Additional information	23
Troubleshooting.....	23
Contacting Technical Support.....	23
 Tutorials	 25
The Example application.....	25
Power BI (Windows only).....	27
Tableau (Windows only).....	27
Microsoft Excel (Windows only).....	28
 Configuring and connecting to data sources.....	 31
Environment settings.....	32
Windows environment variables	32
Linux environment variables.....	32
UTF-16 applications on Linux.....	35
Configuring data sources with the Configuration Manager.....	35
Generating connection strings with the Configuration Manager.....	37
Using a connection string.....	37
Additional configuration methods for Linux.....	38
Configuration through the system information (odbc.ini) file.....	38
DSN-less connections.....	41
File data sources.....	42
Testing connections and queries with the Configuration Manager.....	43
Password Encryption Tool (UNIX/Linux only).....	44

Using a logon dialog box.....	45
OAuth 2.0 authentication.....	45
Register your application on the Azure portal.....	46
Obtain application client information and endpoints.....	47
Obtain access and refresh tokens using the Configuration Manager.....	48
Determine the scope.....	50
Configure the driver to use OAuth 2.0.....	50
Performance considerations.....	54
Using the SQL engine server.....	57
Configuring server mode using the Configuration Manager.....	58
Stopping the SQL engine server using the Configuration Manager.....	59
Configuring the SQL engine server using Java options.....	59
Stopping the SQL engine server.....	61
Configuring Java logging for the SQL engine server.....	61
Connection option descriptions.....	63
Access Token.....	68
Application Using Threads.....	69
Audit Columns.....	69
Authentication Method.....	70
Authorization URI.....	70
Batch Update Chunk Size.....	71
Client ID.....	72
Client Secret.....	72
Create Map.....	73
Data Source Name.....	74
DateTime Zone Shift.....	74
Description.....	75
Enable Login Prompt.....	75
Extended Options.....	76
Fetch Size.....	77
Initialization String.....	78
JVM Arguments.....	78
JVM Classpath.....	79
JVM Path.....	80
Keyword Conflict Suffix.....	81
Log Config File.....	81
Proxy Host.....	82
Proxy Password.....	82
Proxy Port.....	83
Proxy User.....	83
Read Only.....	84

Redirect URI.....	84
Refresh Schema.....	85
Refresh Token.....	85
Report Codepage Conversion Errors.....	86
Schema Map.....	87
Scope.....	88
Server Port Number.....	88
Server Proxy Host.....	89
Server Proxy Password.....	90
Server Proxy Port.....	90
Server Proxy User.....	91
Service URL.....	91
Statement Call Limit.....	92
Statement Call Limit Behavior.....	92
SQL Engine Mode.....	93
SQL Service.....	94
Token URI.....	94
Transaction Mode.....	95
Web Service Compress Data.....	95

Supported SQL statements and extensions.....97

Alter Session (EXT).....	98
Delete.....	99
Explain Plan.....	100
Insert.....	100
Specifying an external ID column.....	101
Refresh Map (EXT).....	102
Select.....	103
Select clause.....	104
Update.....	113
Subqueries.....	114
IN predicate.....	114
EXISTS predicate.....	114
UNIQUE predicate.....	115
Correlated subqueries.....	115
SQL expressions.....	116
Column names.....	117
Literals.....	117
Operators.....	119
Functions.....	123
Conditions.....	123

Introduction to the SharePoint List data model125

ALLCOLUMNSLIST.....	127
DEFAULTCOLUMNSLIST.....	132

Welcome to the Progress DataDirect for ODBC for Microsoft SharePoint Driver

The Progress® DataDirect® for ODBC™ for Microsoft SharePoint™ driver supports SQL read-write access for ODBC applications to list resources on Microsoft SharePoint Online. To support SQL access to SharePoint lists, the driver creates a relational map of the SharePoint data model and translates SQL statements to SharePoint requests. In addition, the driver employs a SQL engine component that provides support to SQL constructs unavailable in Microsoft SharePoint. This functionality offers a number of advantages, including support for reporting data and metadata in a form that ODBC applications are ready to use.

For an overview of the relational tables exposed by the driver and their corresponding API calls, see [Introduction to the SharePoint List data model](#).

The documentation for the driver also includes the *Progress DataDirect for ODBC Drivers Reference*. The reference provides general reference information for all DataDirect drivers for ODBC, including content on troubleshooting, supported SQL escapes, and DataDirect tools. For the complete documentation set, visit to the Progress DataDirect Connectors Documentation Hub: <https://docs.progress.com/bundle/datadirect-connectors/page/DataDirect-Connectors-by-data-source.html>.

For details, see the following topics:

- [What's new in this release?](#)
- [Driver requirements](#)
- [Installing and setting up the driver \(Windows\)](#)
- [Installing and setting up the driver \(Linux\)](#)
- [Connection string examples](#)
- [Data types](#)

- [Driver specifications](#)
- [Additional information](#)
- [Troubleshooting](#)
- [Contacting Technical Support](#)

What's new in this release?

For the latest certifications and enhancements, refer to the following:

- [Release Notes](#)
- [Supported Configurations](#)
- [DataDirect Support Matrices](#)

Visit the following web pages for the latest support and certification information.

- Release Notes: <https://www.progress.com/datadirect-connectors/whats-new#jdbc>
- DataDirect Product Compatibility Guide: <https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf>

Changes Since 8.0.0 GA

- **Enhancements:**
 - The driver is now compiled with a Visual Studio 2022 compiler for the Windows platforms. As a result, you must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher on your machine to run the driver.
 - The driver has been enhanced to support fetching access and refresh tokens at connection when OAuth2.0 is enabled. When using the new dynamic authorization code grant, you can initiate an authorization code grant flow by specifying login credentials using the login prompt for your service, thereby providing a method to authenticate without fetching access and refresh tokens via the Configuration Manager or third-party application. In addition, the new Enable Login Prompt (EnableLoginPrompt) option has been added to enable this functionality. See [Dynamic authorization code grant](#) on page 52 and [Enable Login Prompt](#) on page 75 for details.
 - A Password Encryption Tool, called ddencpwd, is now included with the product package. It encrypts passwords for secure handling in connection strings and odbc.ini files. At connection, the driver decrypts these passwords and passes them to the data source as required. See [Password Encryption Tool \(UNIX/Linux only\)](#) on page 44 for details.

Highlights of 8.0.0 Release

- The driver supports SQL read-write access to list resources on Microsoft SharePoint Online. See [Supported SQL statements and extensions](#) on page 97 for details.
- The driver supports all ODBC Core and Level 1 functions and some Level 1 and Level 2 features.
Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for additional information.
- The driver supports SharePoint data types through data type inference. See [Data types](#) on page 21 and [Introduction to the SharePoint List data model](#) on page 125 for details.

- The driver supports OAuth 2.0 authentication. See [OAuth 2.0 authentication](#) on page 45 for supported grant type and further details.
- The driver supports the handling of large result sets with paging, and the Fetch Size (FetchSize) connection option. See [Fetch Size](#) on page 77 for details.
- The driver supports optimizing insert, update, and delete operations with the [Batch Update Chunk Size](#) on page 71 connection option.
- On Windows platforms, the driver includes an enhanced setup dialog, the DataDirect ODBC Driver Configuration Manager, for quick configuration and testing of your driver. This Configuration Manager allows you to:
 - Configure data sources
 - Generate and edit connection strings
 - Test connect data sources and connection strings
 - Execute SQL commands for testing
 - Access connection option descriptions and the full product documentation

See [Configuring data sources with the Configuration Manager](#) on page 35 and [Generating connection strings with the Configuration Manager](#) on page 37 for details.

Driver requirements

Data source and platform requirements

For the latest support information, visit the DataDirect Product Compatibility Guide:

<https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf>.

Java requirements

- The driver requires a Java Virtual Machine (JVM) that is Java SE 8 or higher. JVM support includes Oracle JDK, OpenJDK, and IBM SDK (Java) distributions.
- For 32-bit drivers, a 32-bit Java Virtual Machine (JVM) is required. For 64-bit drivers, a 64-bit Java Virtual Machine (JVM) is required.
- For Windows, you must set the PATH environment variable to the directory containing the `jvm.dll` for your JVM.
- For Linux, you must set the library path environment variable of your operating system to the directory containing your JVM's `libjvm.so` file and that directory's parent directory.

Windows requirements for 32-bit drivers

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- You must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

Windows requirements for 64-bit drivers

- All required network software that is supplied by your database system vendors must be 64-bit compliant.
- You must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

Linux requirements for 32-bit drivers

- If your application was built with 32-bit system libraries, you must use 32-bit drivers. The database to which you are connecting can be either 32-bit or 64-bit enabled.
- An application compatible with components that were built using g++ GNU project C++ Compiler version 3.4.6 and the Linux native pthread threading model (Linuxthreads).

Linux requirements for 64-bit drivers

- An application compatible with components that were built using g++ GNU project C++ Compiler version 3.4 and the Linux native pthread threading model (Linuxthreads).

Installing and setting up the driver (Windows)

This section provides you with an overview of the steps required to install and set-up the driver. After completing this procedure, you will be able to begin accessing data with your application.

To begin accessing data with the driver:

1. Install the driver:
 - a) After downloading the product, unzip the installer files to a temporary directory.
 - b) From the installer directory, run the appropriate installer file to start the installer. The installer file takes the following form:

```
PROGRESS_DATADIRECT_ODBC_nn_WIN_xx_INSTALL.exe
```

- c) Follow the prompts to complete installation.

Note:

The installer program supports multiple installation methods, including command-line and silent installations. For detailed instructions, refer to the *Progress DataDirect for ODBC Drivers Installation Guide*.

2. Before you can use your driver, you must set the PATH environment variable to include the path of the `jvm.dll` file of your Java™ Virtual Machine (JVM).
3. To configure the driver using the ODBC Administrator (GUI), start the ODBC Administrator from the Progress DataDirect program group. The GUI dialog allows you to configure the data source definitions in the Windows Registry or generate connection strings.

Note: The Windows driver also supports using connection strings to connect to your service. For more information, see "Using a connection string."

4. Select either the **User DSN**, **System DSN**, or **File DSN** tab to display a list of data sources.

- **User DSN:** If you installed a default DataDirect ODBC user data source as part of the installation, select the appropriate data source name and click **Configure** to display the driver Setup dialog box.

If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the driver Setup dialog box.

- **System DSN:** To configure a new system data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the driver Setup dialog box.
- **File DSN:** Configuring a new data source using the File DSN tab is not currently supported with the configuration manager.

5. The **Connection** tab of the Configuration Manager opens in a window. Provide values for the following essential connection options; then, click **Apply**:

For all connections:

- **Data Source Name:** Type a string that identifies this data source configuration, such as `Projects`.
- **Description:** Type an optional long description of a data source name, such as `My Development Projects`.
- **Service URL:** Type the base URL of the SharePoint instance to which you want to issue requests. For example, `https://mycorp.sharepoint.com/sites/marketing/global`.

For OAuth 2.0 access token authentication:

- **Authentication Method:** Determines which authentication method the driver uses during the course of a session. Set to **24 - OAuth2**. This is the default authentication method.
 - **Access Token:** Enter the token required to authenticate to your SharePoint service.
-
- **Important:** The Access Token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.
-

Note: See [OAuth 2.0 authentication](#) on page 45 for details

6. Set the values for any additional connection options that you want to configure. To view more options, click on the tabs on the dialog. See the following resources for additional information on optional features and functionality:

- [Connection string examples](#) provides connection string examples that can be used to configure common functionality and features. The options and values described in this section apply to all configuration methods.
- [Connection option descriptions](#) provides a complete list of supported options by functionality.
- [Configuring data sources with the Configuration Manager](#) guides you through using the GUI to configure the driver.
- [Performance considerations](#) describes connection options that affect performance, along with recommended settings.

Note: For most connections, specifying the minimum required connection options is sufficient to begin accessing data; however, you can provide values for optional connection options to use additional supported features and improve performance.

7. Click **Test Connect** to attempt to connect to the data source using the connection options.

8. If you are using OAuth 2.0 authentication, the logon dialog appears. Update the following fields; then, click **OK**.

- **Client ID:** Enter the client ID key for your application.
- **Client Secret:** Enter the client secret for your application.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

- **Refresh Token:** Enter the refresh token used to either request a new access token or renew an expired access token.
- **Scope:** Enter an OAuth scope or a space-separated list of OAuth scopes that limit the permissions granted by an access token.
- **Authorization URI:** Enter the endpoint for obtaining an authorization code from a third-party authorization service.
- **Token URI:** Enter the endpoint from which the driver fetches access tokens.

Note: The information you enter in the logon dialog box during a test connect is not saved.

9. If the test was successful, the window displays a confirmation message.

10. Click **OK** to close the setup dialog. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the Configuration Manager to modify your data source, or you can override these defaults by connecting to the data source using a connection string with alternate values.

11. Connect to your service and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:

- **Example Application:** The example application allows you to test connect, execute SQL statements, and practice using the ODBC API right out of the box.
- **Power BI:** Power BI is a business intelligence software program that allows you to generate analytics and visualized representations of your data.
- **Tableau:** Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data.
- **Microsoft Excel:** Excel is a spreadsheet tool that allows you to connect, view tables, and execute SQL statements against your data.
- **Supported SQL statements and extensions:** This section describes the syntax used for SQL statements supported by the driver. You can modify and use the provided examples for your application or tool.

This completes the deployment of the driver.

See also

[Using a connection string](#) on page 37

Installing and setting up the driver (Linux)

This section provides you with an overview of the steps required to install and set-up the driver. After completing this procedure, you will be able to begin accessing data with your application.

To begin accessing data with the driver:

1. Install the driver:

- a) After downloading the product, extract the contents of the product file.
- b) From the installer directory, run the installer's binary file to start the installer. The file for the installer program takes the following form:

```
PROGRESS_DATADIRECT_ODBC_nn_LINUX_xx_INSTALL.bin
```

- c) Follow the prompts to complete installation.

The installer program supports multiple installation methods, including command-line and silent installations. For detailed instructions, refer to the *Progress DataDirect for ODBC Drivers Installation Guide*.

2. Configure the environment variables:

- a) Check your permissions. You must log in as a user with full r/w/x permissions recursively on the entire product installation directory.
- b) Run one of the following product setup scripts from the installation directory to set variables: `odbc.sh` or `odbc.csh`. For Korn, Bourne, and equivalent shells, execute `odbc.sh`. For a C shell, execute `odbc.csh`. Executing the setup script:
 - Sets the ODBCINI environment variable to point to the path from the root directory to the system information file where your data source resides. For details, see "ODBCINI."
 - Sets the library path environment variable for your Linux operating system, `LD_LIBRARY_PATH`, to include the directory containing your JVM's `libjvm.so` file. For details, see "Library search path."

3. Configure the driver using one of the following methods:

- **odbc.ini file:** You can begin using the driver immediately by editing the `odbc.ini` file in the installation directory with a text editor. The following demonstrates a data source definition with the minimal options.

OAuth 2.0 refresh token grant

```
[ODBC Data Sources]
Microsoft SharePoint=DataDirect 8.0 Microsoft SharePoint

[Microsoft SharePoint]
Driver=ODBCHOME/lib/xxsharepoint28.yy
...
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global;
...
ClientId=jsmith
...
ClientSecret=secret
...
RefreshToken=hfvdSifuh435th43i5ybg4
...
TokenURI=https://login.microsoftonline.com/common/oauth2/v2.0/token
...
```

See [Configuration through the system information \(odbc.ini\) file](#) on page 38 for more information.

- **Connection string:** The driver also supports using connection strings for DSN (data source name), File DSN, or DSN-less connections. See [Using a connection string](#) on page 37, [DSN-less connections](#), for more information. For examples, see [Connection string examples](#) on page 16.

Note: For most connections, specifying the minimum required connection options is sufficient to begin accessing data; however, you can provide values for optional connection options to use additional supported features and improve performance.

4. Set the values for any additional options that you want to configure. For additional information on optional features and functionality, see the following resources:

- [Connection string examples](#) provides connection string examples that can be used to configure common functionality and features. You can modify and combine these examples to create a string that best suits your environment.

Note: The options and values described in "Connection string examples" apply to all configuration methods.

- [Connection option descriptions](#) provides a complete list of supported options by functionality.
 - [Performance considerations](#) describes connection options that affect performance, along with recommended settings.
5. Connect to your service and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:
 - [Example Application:](#) The example application is a command-line tool that allows you to test connect, execute SQL statements, and practice using the ODBC API in environments that do not support GUIs.
 - [Supported SQL statements and extensions:](#) This section describes the syntax used for SQL statements supported by the driver. You can modify and use the provided examples for your application or tool.

This completes the deployment of the driver.

See also

[Connection string examples](#) on page 16

Connection string examples

ODBC provides a method for specifying connection information via a connection string and the `SQLDriverConnect` API. This section provides examples of connection strings configured to use common features and functionality. You can modify and/or combine these examples to create a connection string for your environment.

In addition to the connection strings for DSN-less connections demonstrated in this section, the driver supports DSN and File DSN connection strings. See "Using a connection string" for syntax and detailed information for supported connection string types.

Note: The options and values described in this section apply to all configuration methods.

See also

[Using a connection string](#) on page 37

OAuth 2.0 access token flow

The access token flow passes the access token directly from the client to the SharePoint service for authentication.

Note: As opposed to using a third-party application such as Postman, you can use the Progress DataDirect SharePoint Configuration Manager to obtain an access token to support the access token flow. See [Obtain access and refresh tokens using the Configuration Manager](#) for details.

Note: Access tokens are temporary and must be replaced to maintain the session without interruption. The life of an access token is typically one hour.

The following string includes the properties used to connect with the OAuth 2.0 access token flow.

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;ServiceURL=serviceurl;  
AccessToken=access_token;[attribute=value[;...]];
```

where:

serviceurl

specifies the URL of a SharePoint site (or "site collection"), for example, `https://mycorp.sharepoint.com/sites/marketing/global`. The URL must include the base URL (`https://mycorp.sharepoint.com/`), as well as the path to the site (`marketing/global`). The driver uses the base URL to infer an OData endpoint, and the site path is used to locate the site that will be accessed.

access_token

specifies the access token required to authenticate to SharePoint. This property allows you to set the access token manually.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties for connecting with the OAuth 2.0 access token flow.

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;ServiceURL=https://mycorp.sharepoint.com  
/sites/marketing/global/;AccessToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

OAuth 2.0 refresh token grant

The refresh token grant is used to replace expired access tokens with active ones by exchanging the refresh token at the endpoint specified by the TokenURI property.

Note: As opposed to using a third-party application such as Postman, you can use the Progress DataDirect SharePoint Configuration Manager to obtain a refresh token to support the refresh token grant. See [Obtain access and refresh tokens using the Configuration Manager](#) for details.

This string includes the properties used to connect with OAuth 2.0 authentication.

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;ServiceURL=serviceurl;ClientID=client_id;
ClientSecret=client_secret;RefreshToken=refresh_token;TokenURI=token_uri;
[property=value[;...]];
```

where:

serviceurl

specifies the URL of a SharePoint site (or "site collection"), for example, `https://mycorp.sharepoint.com/sites/marketing/global`. The URL must include the base URL (`https://mycorp.sharepoint.com/`), as well as the path to the site (`marketing/global`). The driver uses the base URL to infer an OData endpoint, and the site path is used to locate the site that will be accessed.

client_id

specifies the client ID key for your application.

client_secret

specifies the client secret for your application.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

refresh_token

specifies the refresh token used to either request a new access token or renew an expired access token.

Important: The refresh token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

token_uri

specifies the endpoint used to exchange authentication credentials for access tokens.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties required for connecting with the OAuth 2.0 refresh token grant.

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global/;
ClientID=2946f-6789-25gh-gd8g-44t1;ClientSecret=12a3=bCD/E4Ijk+LmqR8s=/TuV+Wbcd;
TokenURI=https://login.microsoftonline.com/common/oauth2/v2.0/token;
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

OAuth 2.0 dynamic authorization code grant

The OAuth 2.0 dynamic authorization code grant allows you to initiate an authorization code grant flow by specifying login credentials using the login prompt for your service, thereby providing a method to authenticate without fetching access and refresh tokens via the Configuration Manager or third-party application.

The following string includes the options used to connect with the OAuth 2.0 dynamic authorization code grant.

```
Driver=DataDirect 8.0 Microsoft SharePoint;
AuthenticationMethod=24;EnableLoginPrompt=true;SQLEngineMode=2;
ServiceURL=serviceurl;ClientID=client_id;ClientSecret=client_secret;
AuthURI=auth_uri;TokenURI=token_uri;Scope=scope;
RedirectURI=redirect_uri;[attribute=value[;...]];
```

where:

serviceurl

specifies the URL of a SharePoint site (or "site collection"), for example, `https://mycorp.sharepoint.com/sites/marketing/global`. The URL must include the base URL (`https://mycorp.sharepoint.com/`), as well as the path to the site (`marketing/global`). The driver uses the base URL to infer an OData endpoint, and the site path is used to locate the site that will be accessed.

client_id

specifies the client ID key for your application.

client_secret

specifies the client secret for your application.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

auth_uri

specifies the endpoint for obtaining an authorization code from a third-party authorization service.

token_uri

specifies the endpoint used to exchange authentication credentials for access tokens.

scope

specifies an OAuth scope or a space-separated list of OAuth scopes that limit the permissions granted by an access token. For example: `Scope=Web.Read List.Write`.

redirect_uri

specifies the endpoint to which the client is returned after authenticating with a third-party service. Note that the value of the Redirect URI (`RedirectURI`) option must include the port number. For example, `RedirectURI=http://localhost:80` or `RedirectURI=http://localhost:8080`.

attribute=value

specifies connection option settings. Multiple options are separated by a semi-colon.

The following example connection string includes the options for connecting with the OAuth 2.0 dynamic authorization code grant.

```
Driver=DataDirect 8.0 Microsoft SharePoint;  
AuthenticationMethod=24;EnableLoginPrompt=true;SQLEngineMode=2;  
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global;  
ClientID=abcdefghijklm3o4p5qr67s;ClientSecret=FaZBFRsGXTaR;  
AuthURI=https://login.microsoftonline.com/xyzabc111000/oauth2/v2.0/authorize;  
TokenURI=https://login.microsoftonline.com/xyz123abc/oauth2/token;  
Scope=https://mycorp.sharepoint.com/offline_access;  
RedirectURI=http://localhost:80;
```

Proxy server

This string includes the options you may need to connect through a proxy server with OAuth 2.0 refresh token grant authentication.

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;ServiceURL=serviceurl;  
ProxyHost=proxy_host;ProxyPassword=proxy_password;ProxyPort=proxy_port;  
ProxyUser=proxy_user;ClientID=client_id;ClientSecret=client_secret;  
TokenURI=token_uri;RefreshToken=refresh_token;[attribute=value[...]];
```

where:

serviceurl

specifies the URL of a SharePoint site (or "site collection"). For example, `https://mycorp.sharepoint.com/sites/marketing/global`. The URL must include the base URL (`https://mycorp.sharepoint.com/`), as well as the path to the site (`marketing/global`). The driver uses the base URL to infer an OData endpoint, and the site path is used to locate the site that will be accessed.

proxy_host

specifies the proxy server to use for the first connection.

proxy_password

specifies the password needed to connect to a proxy server for the first connection.

proxy_port

specifies the port number where the proxy server is listening for requests for the first connection. The default is 0.

proxy_user

specifies the user name needed to connect to a proxy server for the first connection.

client_id

specifies the client ID key for your application when authenticating with OAuth 2.0.

client_secret

specifies the client secret for your application when authenticating with OAuth 2.0.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

token_uri

specifies the endpoint used to exchange authentication credentials for access tokens when OAuth 2.0 authentication is enabled.

refresh_token

specifies the refresh token used to either request a new access token or renew an expired access token..

Important: The refresh token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

attribute=value

specifies connection option settings. Multiple options are separated by a semi-colon.

The following example connection string includes the options required for using a proxy server with OAuth 2.0 authentication.

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global;
ProxyHost=pserver;ProxyPassword=secret;ProxyPort=808;ProxyUser=jsmith;
ClientID=2946f-6789-25gh-gd8g-44t1;ClientSecret=16a3=bED/HfR8s//TuV+WZabcd;
TokenURI=https://login.microsoftonline.com/common/oauth2/v2.0/token;
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

See also

[Connection option descriptions](#) on page 63

Data types

The following table lists native data types supported by the driver and how they are mapped to ODBC data types.

Table 1: Microsoft SharePoint Data Types

Microsoft SharePoint Data Type	ODBC Data Type
BINARY	SQL_VARBINARY
BOOLEAN	SQL_BIT
BYTE	SQL_SMALLINT
DATE	SQL_TYPE_DATE
DATETIMEOFFSET	SQL_TYPE_TIMESTAMP
DECIMAL	SQL_DECIMAL

Microsoft SharePoint Data Type	ODBC Data Type
DOUBLE	SQL_DOUBLE
ENUM	SQL_VARCHAR
GUID	SQL_CHAR
INT16	SQL_SMALLINT
INT32	SQL_INTEGER
INT64	SQL_BIGINT
SBYTE	SQL_TINYINT
SINGLE	SQL_REAL
STRING	SQL_VARCHAR
TIMEOFDAY	SQL_TYPE_TIME

Driver specifications

This section describes the general functionality supported by the driver.

- **ODBC compliance:** The driver is compliant with the Open Database Connectivity (ODBC) 3.52 specification. The driver is ODBC core compliant and supports some Level 1 and Level 2 features.

Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for a list of supported API functions.

The driver supports only the following Level 2 functions:

- SQLColumnPrivileges
 - SQLDescribeParam
 - SQLForeignKeys
 - SQLPrimaryKeys
 - SQLProcedures
 - SQLTablePrivileges
- **Unicode support:** The driver is fully Unicode enabled. On Linux platforms, the driver supports both UTF-8 and UTF-16. On Windows platforms, the driver supports UCS-2/UTF-16 only.
Refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for details.
 - **Isolation and lock levels:** Because transactions are not supported, the driver supports only the isolation level 0 (read uncommitted).

Refer to "Locking and isolation levels" in the *Progress DataDirect for ODBC Drivers Reference* for details.

- **Connections and statements supported:** The driver supports multiple connections and multiple statements per connection.

Additional information

In addition to the content provided in this guide, the documentation set also contains detailed conceptual and reference information that applies to all the drivers. For more information in these topics, refer the *Progress DataDirect for ODBC Drivers Reference* or use the links below to view some common topics:

- "Code page values" lists supported code page values, along with a description, for the Progress DataDirect for ODBC drivers.
- "ODBC API and scalar functions" lists the ODBC API functions supported by Progress DataDirect for ODBC drivers. In addition, it documents the scalar functions that you use in SQL statements.
- "Internationalization, localization, and Unicode" provides an overview of how internationalization, localization, and Unicode relate to each other. It also includes a background on Unicode, and how it is accommodated by Unicode and non-Unicode ODBC drivers.
- "Security best practices for ODBC applications" describes the security best practices you should employ when developing and deploying your application with the driver.

Troubleshooting

The *Progress DataDirect for ODBC Drivers Reference* provides information on troubleshooting problems should they occur.

Refer to the "Troubleshooting" section in the *Progress DataDirect for ODBC Drivers Reference* for details.

Contacting Technical Support

Progress DataDirect offers a variety of options to meet your support needs. Please visit our Web site for more details and for contact information:

<https://www.progress.com/support>

The Progress DataDirect Web site provides the latest support information through our global service network. The SupportLink program provides access to support contact details, tools, patches, and valuable information, including a list of FAQs for each product. In addition, you can search our Knowledgebase for technical bulletins and other information.

When you contact us for assistance, please provide the following information:

- Your number or the serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.
- Your name, phone number, email address, and organization. For a first-time call, you may be asked for full information, including location.

- The Progress DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your product.
- Any database, database version, third-party software, or other environment information required to understand the problem.
- A brief description of the problem, including, but not limited to, any error messages you have received, what steps you followed prior to the initial occurrence of the problem, any trace logs capturing the issue, and so on. Depending on the complexity of the problem, you may be asked to submit an example or reproducible application so that the issue can be re-created.
- A description of what you have attempted to resolve the issue. If you have researched your issue on Web search engines, our Knowledgebase, or have tested additional configurations, applications, or other vendor products, you will want to carefully note everything you have already attempted.
- A simple assessment of how the severity of the issue is impacting your organization.

December 2020, 8.0.0 Release of Progress DataDirect for ODBC for Microsoft SharePoint, Version 0001

Tutorials

The following sections guide you through using the driver to access your data with some common third-party applications. For information on installing your driver and setting the CLASSPATH, see "Installing and setting-up the driver (Windows)" or "Installing and setting-up the driver (Linux)."

For details, see the following topics:

- [The Example application](#)
- [Power BI \(Windows only\)](#)
- [Tableau \(Windows only\)](#)
- [Microsoft Excel \(Windows only\)](#)

The Example application

The driver installation includes an ODBC application called Example that can be used for:

- Testing any type of SQL statement
- Testing database connections
- Verifying your database environment

It can also be used to demonstrate ODBC function calls, including the following:

- SQLAllocHandle
- SQLBindCol
- SQLBindParameter
- SQLColAttribute
- SQLConnect
- SQLDescribeCol
- SQLDescribeParam
- SQLDisconnect
- SQLDriverConnect
- SQLExecDirect
- SQLFetch
- SQLFreeHandle
- SQLFreeStmt
- SQLGetDiagRec
- SQLGetInfo
- SQLNumResultCols
- SQLPrepare
- SQLSetEnvAttr
- SQLSetStmtAttr

The Example application can be built using the files located in the `\samples\examples` directory of the DataDirect for ODBC Drivers installation directory.

Note:

- For Windows, you can build the Windows app for ANSI and Unicode.
- For Linux, instructions for building the Example application are contained inside the file `example.mak`, which can be read with a text editor.

To use the Example application:

1. After you have configured the data source, navigate to the `instal_dir\samples\example` directory.
2. Open the application using one of the following methods:

- Running the application executable or binary:
 - On Windows, double-click the `Example.exe` file.
 - On Linux, run the `example` application.
- Executing a command-line argument. For example:
 - `example connection_string`
 - `example "DSN" "UID" "PWD"`
 - `example connection_string "sql_command_1" ["sql_command_2" ...]`

Results: A command prompt opens.

3. Follow the prompts to enter your data source name, user name, and password. If successful, a `SQL>` prompt appears.
4. At the prompt, enter SQL statements to test your connection. For example:

```
SELECT * FROM INFORMATION_SCHEMA.TABLES
```

The results of your query are displayed. If `example` is unable to connect, the appropriate error message is returned.

Power BI (Windows only)

After you have configured your data source, you can use the driver to access your data with Power BI. Power BI is a business intelligence software program that allows you to easily create reports and visualized representations of your data. By using the driver with Power BI, you can improve performance when retrieving data while leveraging the driver's relational mapping tools.

1. Navigate to the `\tools\Power BI` subdirectory of the Progress DataDirect installation directory; then, locate the installation batch file `install.bat`.
2. Run the `install.bat` file. The following operations are executed by running the `install.bat` file:
 - The Power BI connector file, `DataDirectSharePoint.pqx`, is copied to the following directory.
`%USERPROFILE%\Documents\Power BI Desktop\Custom Connectors`
 - The following Windows registry entry is updated.
`HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Power BI Desktop\TrustedCertificateThumbprints`
3. Open the Power BI desktop application.
4. From the **Get Data** window, navigate to **Other > Progress DataDirect Microsoft SharePoint Connector**.
5. Click **Connect**. Then, from the **Progress DataDirect Microsoft SharePoint Connector** window, provide the following information. Then, click **OK**.
 - **Data Source:** Enter a name for the data source. For example, `Microsoft SharePoint ODBC DSN`.
 - **SQL Statement:** If desired, provide a SQL command.
 - **Data Connectivity mode:**
 - Select **Import** to import data to Power BI.
 - Select **DirectQuery** to query live data. (For details, including limitations, refer to the Microsoft Power BI article [Use DirectQuery in Power BI Desktop](#).)
6. Enter authentication information when prompted. Once connected, the **Navigator** window displays schema and table information.
7. Select and load tables. Then, prepare your Power BI dashboard as desired.

You have successfully accessed your data and are now ready to create reports with Power BI. For more information, refer to the Power BI product documentation at [Power BI documentation](#).

Tableau (Windows only)

After you have configured your data source, you can use the driver to access your data with Tableau. Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data. By using the driver with Tableau, you can improve performance when retrieving data while leveraging the driver's relational mapping tools.


To use the driver to access data with Tableau:

1. Navigate to the `\tools\Tableau` subdirectory of the Progress DataDirect installation directory; then, locate the following Tableau data source file:

```
DataDirect SharePoint.tdc
```

2. Copy the Tableau data source file into the following directory:

```
C:\Users\user_name\Documents\My Tableau Repository\Datasources
```

3. Open Tableau. If the **Connect** menu does not open by default, select **Data > New Data Source** or the Add New Data Source button  to open the menu.
4. From the **Connect** menu, select **Other Databases (ODBC)**.
5. The **Other Databases (ODBC)** dialog appears. In the DSN field, select the data source you want to use from the drop-down menu. For example, **My DSN**. Then, click **Connect**. The Logon dialog appears pre-populated with the connection information you provided in your data source.
6. If required, type your user name and password; then, click **OK**. The Logon dialog closes. Then, click **Sign in** on the Other Databases (ODBC) dialog.
7. The **Data Source** window appears. By default, Tableau connects live, or directly, to your data. We recommend that you use the default settings to avoid extracting all of your data. However, if you prefer, you can import your data by selecting the **Extract** option at the top of the dialog.
8. In the Schema field, select the schema you want to use. The tables stored in this schema are now available for selection in the Table field.

You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: <http://www.tableau.com/support/help>.

Microsoft Excel (Windows only)

After you have configured your data source, you can use the driver to access your data with Microsoft Excel from the Data Connection Wizard. Using the driver with Excel provides improved performance when retrieving data, while leveraging the driver's relational-mapping tools.

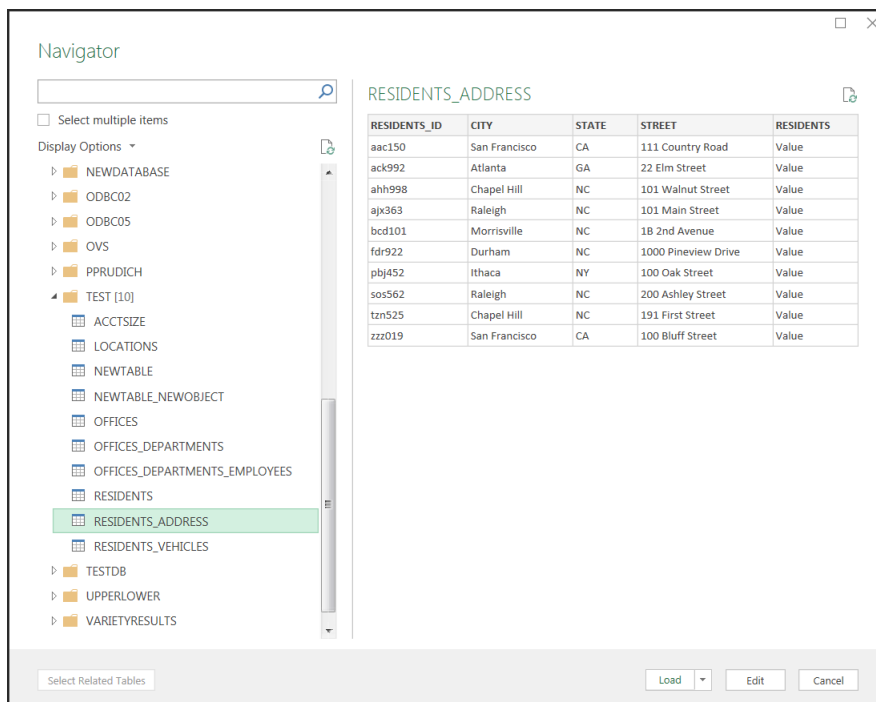
To use the driver to access data with Excel from the Data Connection Wizard:

1. Open your workbook in Excel.
2. From the **Data** menu, select **Get Data>From Other Sources>From ODBC**.
3. The **From ODBC** dialog appears.



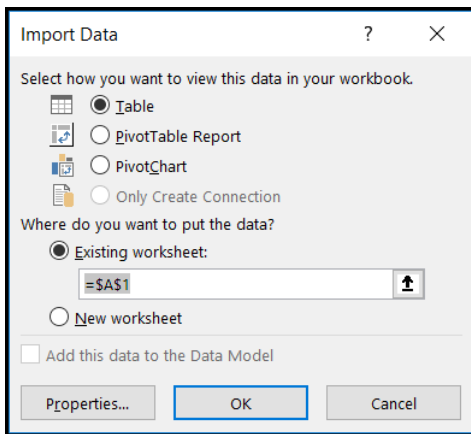
Select your data source from the Data Source Name (DSN) drop down; then, click **OK**.

4. You are prompted for logon credentials for your data source:
 - If your data source does not require logon credentials or if you prefer to specify your credentials using a connection string, select **Default or Custom** from the menu on the left. Optionally, specify your credential-related options using a connection string in the provided field. Click **Connect** to proceed.
 - If your data source uses Windows credentials, select **Windows** from the menu; then, provide your credentials. Optionally, specify a connection string with credential-related options in the provided field. Click **Connect** to proceed.
 - If your data source uses credentials stored on the database, select **Database**; then, provide your user name and password. Optionally, specify a connection string in the provided field. Click **Connect** to proceed.
5. The **Navigator** window appears.



From the list, select the tables you want to access. A preview of your data will appear in the pane on the right. Optionally, click **Edit** to modify the results using the Query Editor. Refer to the Microsoft Excel product documentation for detailed information on using the Query Editor.

6. Load your data:
 - Click **Load** to import your data into your work sheet. Skip to the end.
 - Click **Load>Load To** to specify a location to import your data. Proceed to the next step.
7. The **Import Data** window appears.



Select the desired view and insertion point for the data. Click **OK**.

You have successfully accessed your data in Excel. For more information, refer to the Microsoft Excel product documentation at: <https://support.office.com/>.

Configuring and connecting to data sources

After you install the driver, you configure data sources to connect to the database. Data sources store the information that the driver needs to connect to a database. The ODBC specification describes three types of data sources: user data sources, system data sources (not a valid type on Linux), and file data sources. On Windows, user and system data sources are stored in the registry of the local computer. The difference is that only a specific user can access user data sources, whereas any user of the machine can access system data sources. On Windows and Linux, file data sources, which are simply text files, can be stored locally or on a network computer, and are accessible to other machines.

The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Connection option descriptions" for an alphabetical list of driver connection string attributes and their initial default values.

For details, see the following topics:

- [Environment settings](#)
- [Configuring data sources with the Configuration Manager](#)
- [Generating connection strings with the Configuration Manager](#)
- [Using a connection string](#)
- [Additional configuration methods for Linux](#)
- [Testing connections and queries with the Configuration Manager](#)
- [Password Encryption Tool \(UNIX/Linux only\)](#)
- [Using a logon dialog box](#)

- [OAuth 2.0 authentication](#)
- [Performance considerations](#)

Environment settings

The first step in setting up and configuring the driver for use is to set environment settings and variables. The following procedures require that you have the appropriate permissions to modify your environment and to read, write, and execute various files. You must log in as a user with full r/w/x permissions recursively on the entire Progress DataDirect for ODBC installation directory.

Windows environment variables

Before you can use your driver, you must set the PATH environment variable to include the path of the `jvm.dll` file of your Java™ Virtual Machine (JVM).

Note: During installation, the Windows installer sets the PATH environment variable to include the path of the JVM.

Linux environment variables

The following topics guide you through setting the environment variables for Linux. You must set these environment variables before connecting with your driver.

Library search path

The library search path variable can be set by executing the appropriate shell script located in the ODBC home directory. From your login shell, determine which shell you are running by executing:

```
echo $SHELL
```

C shell login (and related shell) users must execute the following command before attempting to use ODBC-enabled applications:

```
source ./odbc.csh
```

Bourne shell login (and related shell) users must initialize their environment as follows:

```
. ./odbc.sh
```

Executing these scripts sets the library search path environment variable, `LD_LIBRARY_PATH`.

The library search path environment variable must be set so that the ODBC core components, Java components, and drivers can be located at the time of execution. After running the setup script, execute:

```
env
```

to verify that the `installation_directory/lib` directory has been added to your shared library path.

ODBCINI

The product installer places a default system information file, named `odbc.ini`, that contains data sources in the product installation directory. See "Configuration through the system information (`odbc.ini`) file" for an explanation of the `odbc.ini` file. The system administrator can choose to rename the file and/or move it to another location. In either case, the environment variable `ODBCINI` must be set to point to the fully qualified path name of the `odbc.ini` file.

For example, to point to the location of the file for an installation on `/opt/odbc` in the C shell, you would set this variable as follows:

```
setenv ODBCINI /opt/odbc/odbc.ini
```

In the Bourne or Korn shell, you would set it as:

```
ODBCINI=/opt/odbc/odbc.ini;export ODBCINI
```

As an alternative, you can choose to make the `odbc.ini` file a hidden file and not set the `ODBCINI` variable. In this case, you would need to rename the file to `.odbc.ini` (to make it a hidden file) and move it to the user's `$HOME` directory.

The driver searches for the location of the `odbc.ini` file as follows:

1. The driver checks the `ODBCINI` variable
2. The driver checks `$HOME` for `.odbc.ini`

If the driver does not locate the system information file, it returns an error.

See also

[Configuration through the system information \(`odbc.ini`\) file](#) on page 38

ODBCINST

The installer program places a default file, named `odbcinst.ini`, for use with DSN-less connections in the product installation directory. See "DSN-less connections" for an explanation of the `odbcinst.ini` file. The system administrator can choose to rename the file or move it to another location. In either case, the environment variable `ODBCINST` must be set to point to the fully qualified path name of the `odbcinst.ini` file.

For example, to point to the location of the file for an installation on `/opt/odbc` in the C shell, you would set this variable as follows:

```
setenv ODBCINST /opt/odbc/odbcinst.ini
```

In the Bourne or Korn shell, you would set it as:

```
ODBCINST=/opt/odbc/odbcinst.ini;export ODBCINST
```

As an alternative, you can choose to make the `odbcinst.ini` file a hidden file and not set the `ODBCINST` variable. In this case, you would need to rename the file to `.odbcinst.ini` (to make it a hidden file) and move it to the user's `$HOME` directory.

The driver searches for the location of the `odbcinst.ini` file as follows:

1. The driver checks the `ODBCINST` variable
2. The driver checks `$HOME` for `.odbcinst.ini`

If the driver does not locate the `odbcinst.ini` file, it returns an error.

See also

[DSN-less connections](#) on page 41

DD_INSTALLDIR

This variable provides the driver with the location of the product installation directory so that it can access support files. `DD_INSTALLDIR` must be set to point to the fully qualified path name of the installation directory.

For example, to point to the location of the directory for an installation on `/opt/odbc` in the C shell, you would set this variable as follows:

```
setenv DD_INSTALLDIR /opt/odbc
```

In the Bourne or Korn shell, you would set it as:

```
DD_INSTALLDIR=/opt/odbc;export DD_INSTALLDIR
```

The driver searches for the location of the installation directory as follows:

1. The driver checks the `DD_INSTALLDIR` variable
2. The driver checks the `odbc.ini` or the `odbcinst.ini` files for the `InstallDir` keyword (see "Configuration through the system information (`odbc.ini`) file" for a description of the `InstallDir` keyword)

If the driver does not locate the installation directory, it returns an error.

The next step is to test load the driver.

See also

[Configuration through the system information \(`odbc.ini`\) file](#) on page 38

The test loading tool

The second step in preparing to use a driver is to test load it.

The `ivtestlib` (32-bit driver) and `ddtestlib` (64-bit driver) test loading tools are provided to test load drivers and help diagnose configuration problems in the Linux environment, such as environment variables not correctly set or missing database client components. This tool is installed in the `/bin` subdirectory in the product installation directory. It attempts to load a specified ODBC driver and prints out all available error information if the load fails.

For example, if the driver is installed in `/opt/odbc/lib`, the following command attempts to load the 32-bit driver on Linux, where `xx` represents the version number of the driver:

```
ivtestlib /opt/odbc/lib/ivsharepointxx.so
```

Note: The full path to the driver does not have to be specified for the tool.

If the load is successful, the tool returns a success message along with the version string of the driver. If the driver cannot be loaded, the tool returns an error message explaining why.

The next step is to configure a data source through the system information file.

UTF-16 applications on Linux

Because the DataDirect Driver Manager allows applications to use either UTF-8 or UTF-16 Unicode encoding, applications written in UTF-16 for Windows platforms can also be used on Linux platforms.

The Driver Manager assumes a default of UTF-8 applications; therefore, two things must occur for it to determine that the application is UTF-16:

- The definition of SQLWCHAR in the ODBC header files must be switched from "char *" to "short *". To do this, the application uses #define SQLWCHARSHORT.
- The application must set the encoding for the environment or connection using one of the following attributes. If your application passes UTF-8 encoded strings to some connections and UTF-16 encoded strings to other connections in the same environment, encoding should be set for the connection only; otherwise, either method can be used.

- To configure the encoding for the environment, set the ODBC environment attribute SQL_ATTR_APP_UNICODE_TYPE to a value of SQL_DD_CP_UTF16, for example:

```
rc = SQLSetEnvAttr(*henv, SQL_ATTR_APP_UNICODE_TYPE,
(SQLPOINTER)SQL_DD_CP_UTF16, SQL_IS_INTEGER);
```


- To configure the encoding for the connection only, set the ODBC connection attribute SQL_ATTR_APP_UNICODE_TYPE to a value of SQL_DD_CP_UTF16. For example:

```
rc = SQLSetConnectAttr(hdbc, SQL_ATTR_APP_UNICODE_TYPE, SQL_DD_CP_UTF16,
SQL_IS_INTEGER);
```

Configuring data sources with the Configuration Manager

Note: The Configuration Manager is currently supported only on Windows platforms.

The driver includes an enhanced setup dialog, the Progress DataDirect Microsoft SharePoint Configuration Manager, that allows you to configure data sources, generate connection strings, test connections, and execute test queries. On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using the Configuration Manager, as described in this section.

Note: As you configure your data source, the Configuration Manager generates a corresponding connection string in the **Connection String** pane. To use your connection string, click the Copy button () and paste the string to a location that can be used by your application. See "Generating connection strings with the Configuration Manager" for details.

Note: Connection string attributes can be used to override the default values of the data source if you want to change these values at connection time.

To configure and test a data source:

1. Open the Windows ODBC Administrator.
 2. Open the Configuration Manager through the **User DSN** or **System DSN** tab.
 - **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.
If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the Configuration Manager.
 - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.
If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the Configuration Manager.
-
- **Note**: Configuring a new data source using the File DSN tab is not currently supported with the configuration manager.
-

The Microsoft SharePoint Configuration Manager window opens.

3. From the Configuration Manager window, provide values of the connection options you want to configure in the corresponding fields. To view more options, select the tabs at the top of the page. See "Connection option descriptions" for descriptions of the supported options.

Note: See "Connection string examples" for a list of required options used for different configurations. The options and settings described in that section apply to all methods of configuration.

4. At any point during the process, you can click **Test Connect** to attempt to connect to the service with your settings. In the Test Connection window:
 - a) Provide values for any fields required by your service. Note that the information you enter in the logon dialog box during a test connect is not saved.
 - b) Optionally, in the **Test Query** field, enter any SQL queries you want to execute during the test. For example:

```
SELECT * FROM INFORMATION_SCHEMA.SYSTEM_TABLES
```
 - c) Click **Execute**.

Important: An initial connection may take a few minutes, depending on network speeds and the amount of metadata the driver must retrieve from the service. Similar delays may occur depending on the Create Map (CreateMap) setting. For example, a delay may be incurred if Create Map is set to 4 (OnChange), the default, and changes have been made to the schema on the Microsoft SharePoint backend. For details, see "Create Map".

If successful, the window displays a confirmation message and, if a query was specified, the results of the query.

5. Click **Save** to apply your values as the default when connecting with the data source.

See also

[Generating connection strings with the Configuration Manager](#) on page 37

[Connection string examples](#) on page 16

[Configuration through the system information \(odbc.ini\) file](#) on page 38

Generating connection strings with the Configuration Manager

Note: The Configuration Manager is currently supported only on Windows platforms.

The Progress DataDirect Microsoft SharePoint Configuration Manager supports generating connection strings that can be used with your application. To generate a connection string, create a data source as described in "Configuring data sources with the Configuration Manager." As you provide connection option values, the Configuration Manager generates a connection string in the **Connection String** pane that corresponds to the data source.

In addition to providing values for connection option fields, you can manually edit your string by clicking the Edit button (✎). Note that editing your connection string also changes the values for the data source.

After you are done configuring the connection options, click **Test Connect** to test your connection string. See "Testing connections and queries with the Configuration Manager" for more information.

To use your string, click the Copy button (📄) and paste the string to a location that can be used by your application.

See also

[Configuring data sources with the Configuration Manager](#) on page 35

[Testing connections and queries with the Configuration Manager](#) on page 43

Using a connection string

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the `DSN=`, `FILEDSN=`, or the `DRIVER=` keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because the information is not stored in a data source.

The DSN-less connection string has the form:

```
DRIVER=[{ }driver_name{ }][;attribute=value[;attribute=value]...]
```

"Connection option descriptions" lists the long and short names for each attribute, as well as the initial default value when the driver is first installed. You can specify either long or short names in the connection string.

An example of a DSN connection string with overriding attribute values for the driver for Linux or Windows is:

```
DSN=SharePoint;ClientSecret=12a3=bCD/E4Ijk+LmqR8s>//TuV+Wbcd;
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=SharePoint.dsn;ClientSecret=12a3=bCD/E4Ijk+LmqR8s>//TuV+Wbcd;
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;  
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global/  
AuthenticationMethod=OAuth2;ClientID=2946f-6789-25gh-gd8g-44t1;  
ClientSecret=12a3=bCD/E4Ijk+LmqR8s>//TuV+Wbcd;  
TokenURI=https://login.microsoftonline.com/common/oauth2/v2.0/token;  
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

See also

[Connection option descriptions](#) on page 63

[Connection string examples](#) on page 16

Additional configuration methods for Linux

This section contains configuration methods that are specific to the Linux environment.

Configuration through the system information (odbc.ini) file

In the Linux environments, a system information file is used to store data source information. Setup installs a default version of this file, called `odbc.ini`, in the product installation directory. This is a plain text file that contains data source definitions.

To configure a data source manually, you edit the `odbc.ini` file with a text editor. The content of this file is divided into three sections.

Note: The driver and driver manager support ASCII and UTF-8 encoding in the `odbc.ini` file.

Refer to the "Character encoding in the `odbc.ini` and `odbcinst.ini` files" in *Progress DataDirect for ODBC Drivers Reference* for details.

At the beginning of the file is a section named `[ODBC Data Sources]` containing `data_source_name=installed-driver` pairs, for example:

```
SharePoint2=DataDirect 8.0 Microsoft SharePoint
```

The driver uses this section to match a data source to the appropriate installed driver.

The `[ODBC Data Sources]` section also includes data source definitions. The default `odbc.ini` contains a data source definition for the driver. Each data source definition begins with a data source name in square brackets, for example, `[SharePoint2]`. The data source definitions contain connection string *attribute=value* pairs with default values. You can modify these values as appropriate for your system. "Connection option descriptions" describes these attributes. See "Sample default `odbc.ini` file" for sample data sources.

The second section of the file is named [ODBC File DSN] and includes one keyword:

```
[ODBC File DSN]
DefaultDSNDir=
```

This keyword defines the path of the default location for file data sources (see "File data sources").

Note: This section is not included in the default `odbc.ini` file that is installed by the product installer. You must add this section manually.

The third section of the file is named [ODBC] and includes several keywords, for example:

```
[ODBC]
InstallDir=/opt/odbc
Trace=0
TraceFile=odbctrace.out
TraceDll=/opt/odbc/lib/ivtrc28.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10
```

For supported code page values, refer to "Code page values" in the *Progress DataDirect for ODBC Drivers Reference*.

The `InstallDir` keyword must be included in this section. The value of this keyword is the path to the installation directory under which the `/lib` and `/locale` directories are contained. The installation process automatically writes your installation directory to the default `odbc.ini` file.

For example, if you choose an installation location of `/opt/odbc`, then the following line is written to the [ODBC] section of the default `odbc.ini`:

```
InstallDir=/opt/odbc
```

Note: If you are using only DSN-less connections through an `odbcinst.ini` file and do not have an `odbc.ini` file, then you must provide [ODBC] section information in the [ODBC] section of the `odbcinst.ini` file. The driver and Driver Manager always check first in the [ODBC] section of an `odbc.ini` file. If no `odbc.ini` file exists or if the `odbc.ini` file does not contain an [ODBC] section, they check for an [ODBC] section in the `odbcinst.ini` file. See "DSN-less connections" for details.

ODBC tracing allows you to trace calls to the ODBC driver and create a log of the traces for troubleshooting purposes. The following keywords all control tracing: `Trace`, `TraceFile`, `TraceDLL`, `ODBCTraceMaxFileSize`, and `ODBCTraceMaxNumFiles`.

For a complete discussion of tracing, refer to "ODBC trace" in the *Progress DataDirect for ODBC Drivers Reference*.

See also

[Sample default odbc.ini file](#) on page 40

[File data sources](#) on page 42

[DSN-less connections](#) on page 41

Sample default odbc.ini file

The following is a sample `odbc.ini` file that the installer program installs in the installation directory. All occurrences of `ODBCHOME` are replaced with your installation directory path during installation of the file. Values that you must supply are enclosed by angle brackets (< >). If you are using the installed `odbc.ini` file, you must supply the values and remove the angle brackets before that data source section will operate properly. Commented lines are denoted by the `#` symbol. This sample shows a 32-bit driver with the driver file name beginning with `iv`. A 64-bit driver file would be identical except that driver name would begin with `dd` and the list of data sources would include only the 64-bit drivers.

```
[ODBC Data Sources]
Microsoft SharePoint=DataDirect 8.0 Microsoft SharePoint

[Microsoft SharePoint]
Driver=ODBCHOME/lib/ivsharepoint28.so
Description=DataDirect 8.0 Microsoft SharePoint Driver
AccessToken=
ApplicationUsingThreads=1
AuditColumns=0
AuthenticationMethod=24
AuthURI=
BatchUpdateChunkSize=200
ClientID=
ClientSecret=
CreateMap=0
DateTimeZoneShift=0
FetchSize=100
InitializationString=
JVMArgs=-Xmx256m
JVMClasspath=
KeywordConflictSuffix=
LogConfigFile=ddlogging.properties
ProxyHost=
ProxyPassword=
ProxyPort=0
ProxyUser=
ReadOnly=false
RefreshSchema=0
RefreshToken=
ReportCodepageConversionErrors=0
SchemaMap=
Scope=
ServiceURL=
ServerPortNumber=19963
SQLEngineMode=2
StmtCallLimit=0
StmtCallLimitBehavior=1
TokenURI=
TransactionMode=0
WSCompressData=0

[ODBC]
InstallDir=ODBCHOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBCHOME/lib/ivtrc28.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10

[ODBC File DSN]
DefaultDSNDir=
UseCursorLib=0
```

To modify or create data sources in the `odbc.ini` file, use the following procedures.

- **To modify a data source:**

- a) Using a text editor, open the `odbc.ini` file.
- b) Modify the default values for attributes in the data source definitions as necessary based on your system specifics.
See "Connection option descriptions" for other specific attribute values.
- c) After making all modifications, save the `odbc.ini` file and close the text editor.

Important: The "Connection option descriptions" section lists both the long and short names of the attribute. When entering attribute names into `odbc.ini`, you must use the long name of the attribute. The short name is not valid in the `odbc.ini` file.

- **To create a new data source:**

- a) Using a text editor, open the `odbc.ini` file.
- b) Copy an appropriate existing default data source definition and paste it to another location in the file.
- c) Change the data source name in the copied data source definition to a new name. The data source name is between square brackets at the beginning of the definition, for example, `[My Datasource]`.
- d) Modify the attributes in the new definition as necessary based on your system specifics.
See "Connection option descriptions" for other specific attribute values.
- e) In the `[ODBC]` section at the beginning of the file, add a new `data_source_name=installed-driver` pair containing the new data source name and the appropriate installed driver name.
- f) After making all modifications, save the `odbc.ini` file and close the text editor.

Important: The "Connection option descriptions" section lists both the long and short name of the attribute. When entering attribute names into `odbc.ini`, you must use the long name of the attribute. The short name is not valid in the `odbc.ini` file.

DSN-less connections

Connections to a data source can be made via a connection string without referring to a data source name (DSN-less connections). This is done by specifying the "DRIVER=" keyword instead of the "DSN=" keyword in a connection string, as outlined in the ODBC specification. A file named `odbcinst.ini` must exist when the driver encounters `DRIVER=` in a connection string.

Setup installs a default version of this file in the product installation directory (see "ODBCINST" for details about relocating and renaming this file). This is a plain text file that contains default DSN-less connection information. You should not normally need to edit this file. The content of this file is divided into several sections.

Note: The driver and driver manager support ASCII and UTF-8 encoding in the `odbcinst.ini` file.

Refer to the "Character encoding in the `odbc.ini` and `odbcinst.ini` files" in *Progress DataDirect for ODBC Drivers Reference* for details.

At the beginning of the file is a section named `[ODBC Drivers]` that lists installed drivers, for example,

```
DataDirect 8.0 Microsoft SharePoint=Installed
```

This section also includes additional information for each driver.

The final section of the file is named [ODBC]. The [ODBC] section in the `odbcinst.ini` file fulfills the same purpose in DSN-less connections as the [ODBC] section in the `odbc.ini` file does for data source connections. See "Configuration through the system information (odbc.ini) file" for a description of the other keywords this section.

Note: The `odbcinst.ini` file and the `odbc.ini` file include an [ODBC] section. If the information in these two sections is not the same, the values in the `odbc.ini` [ODBC] section override those of the `odbcinst.ini` [ODBC] section.

See also

[Configuration through the system information \(odbc.ini\) file](#) on page 38

[ODBCINST](#) on page 33

Sample odbcinst.ini file

The following is a sample `odbcinst.ini`. All occurrences of ODBC_HOME are replaced with your installation directory path during installation of the file. Commented lines are denoted by the # symbol. This sample shows a 32-bit driver with the driver file name beginning with `iv`; a 64-bit driver file would be identical except that driver names would begin with `dd`.

```
[ODBC Drivers]
DataDirect 8.0 Microsoft SharePoint=Installed

[DataDirect 8.0 Microsoft SharePoint]
Driver=ODBC_HOME/lib/ivsharepoint28.so
JarFile=ODBC_HOME/java/lib/sharepoint.jar
APILevel=0
ConnectFunctions=YYY
CPTimeout=60
DriverODBCVer=3.52
FileUsage=0
SQLLevel=0
UsageCount=1

[ODBC]
#This section must contain values for DSN-less connections
#if no odbc.ini file exists. If an odbc.ini file exists,
#the values from that [ODBC] section are used.

InstallDir=ODBC_HOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBC_HOME/lib/ivtrc28.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10
```

File data sources

The Driver Manager on Linux supports file data sources. The advantage of a file data source is that it can be stored on a server and accessed by other machines, either Windows or Linux. See "Configuring and connecting to data sources" for a general description of ODBC data sources on both Windows and Linux.

A file data source is simply a text file that contains connection information. It can be created with a text editor. The file normally has an extension of `.dsn`.

For example, a file data source for the driver would be similar to the following:

```
[ODBC]
Driver=DataDirect 8.0 Microsoft SharePoint
...
ServiceURL=http://mycorp.sharepoint.com/sites/marketing/global
...
AccessToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831
...
```

It must contain all basic connection information plus any optional attributes. Because it uses the `DRIVER=` keyword, an `odbcinst.ini` file containing the driver location must exist (see "DSN-less connections").

The file data source is accessed by specifying the `FILEDSN=` instead of the `DSN=` keyword in a connection string, as outlined in the ODBC specification. The complete path to the file data source can be specified in the syntax that is normal for the machine on which the file is located. For example, on Windows:

```
FILEDSN=C:\Program Files\Common Files\ODBC\DataSources\SharePoint2.dsn
```

or, on Linux:

```
FILEDSN=/home/users/john/filedsn/SharePoint2.dsn
```

If no path is specified for the file data source, the Driver Manager uses the `DefaultDSNDir` property, which is defined in the `[ODBC File DSN]` setting in the `odbc.ini` file to locate file data sources (see "Configuration through the system information (odbc.ini) file" for details). If the `[ODBC File DSN]` setting is not defined, the Driver Manager uses the `InstallDir` setting in the `[ODBC]` section of the `odbc.ini` file. The Driver Manager does not support the `SQLReadFileDSN` and `SQLWriteFileDSN` functions.

As with any connection string, you can specify attributes to override the default values in the data source:

```
FILEDSN=/home/users/john/filedsn/SharePoint2.dsn
```

See also

[Configuring and connecting to data sources](#) on page 31

[DSN-less connections](#) on page 41

[Configuration through the system information \(odbc.ini\) file](#) on page 38

Testing connections and queries with the Configuration Manager


Note: The Configuration Manager is currently supported only on Windows platforms.

You can quickly test data sources, connection strings and queries using Progress DataDirect Microsoft SharePoint Configuration Manager.

To test your connection and query:

1. Open the Windows ODBC Administrator. Then, select or add a data source to open the Microsoft SharePoint Configuration Manager.
 - Windows: Start the ODBC Administrator by selecting its icon from the Progress DataDirect program group. To use an existing data source, select one to configure from the ODBC Administrator; otherwise, add a new data source to test new configurations or connection strings.

For detailed information on launching the Configuration Manager, see "Configuring data sources with the Configuration Manager."

2. If you are not testing an existing data source, provide connection information using one of the following methods:
 - Enter a connection string you provide by clicking the Edit button (); then, pasting your string into the Connection String field. If you prefer, you can also type a string directly into this field.
 - Enter values of the connection options you want to configure into the corresponding fields. The Microsoft SharePoint Configuration Manager will generate a data source and connection string based on the values you specify.
3. Click **Test Connect** to attempt to connect to the service using the string specified in the Connection String field. The **Test Connection** window appears.
4. Provide connection option values for any fields required by your service.
5. Optionally, to execute a test query with the test connection, enter a SQL query into the Test Query field. For example:

```
SELECT * FROM INFORMATION_SCHEMA.SYSTEM_TABLES
```

6. Click **Execute**.

Important: An initial connection may take a few minutes, depending on network speeds and the amount of metadata the driver must retrieve from the service.

If successful, the window displays a confirmation message and, if a query was specified, the results of the query.

See also

[Configuring data sources with the Configuration Manager](#) on page 35

Password Encryption Tool (UNIX/Linux only)

On UNIX and Linux, Progress DataDirect provides a Password Encryption Tool, called `ddencpwd`, that encrypts passwords for secure handling in connection strings and `odbc.ini` files. At connection, the driver decrypts these passwords and passes them to the data source as required. Passwords can be encrypted for any option, including:

- KeyPassword
- KeyStorePassword
- TrustStorePassword
- Password

To use the Password Encryption Tool:

1. From a command line, navigate to the directory containing the `ddencpwd` application. By default, this is `install_directory/tools`.

2. Enter the following command:

```
ddencpwd password
```

where:

```
password
```

is the password you want to encrypt.

3. The tool returns an encrypted password value. Specify the returned value for the corresponding attribute in the connection string or `odbc.ini` file. For example, if you encrypted the password for `KeyPassword`, specify the following in your connection string or datasource definition:

```
KeyPassword=returned_value
```

4. Repeat Steps 2 and 3 to encrypt additional passwords.

5. If using an `odbc.ini` file, save your file.

This completes this tutorial. You are now ready to connect using encrypted passwords.

Using a logon dialog box

Some ODBC applications display a logon dialog box when you are connecting to a data source. To connect, provide the values described in the following sections; then, click **OK** to complete the logon.

In the dialog box, provide the following information for OAuth 2.0 authentication method:

- In the Client ID field, type the client ID key for your application.
- In the Client Secret field, type the client secret for your application.
- In the AccessToken field, type the access token used to authenticate to your service.
- In the Refresh Token field, type the refresh token used to connect to your service.
- In the Scope field, type a space-separated list of OAuth scopes.
- In the Authorization URI field, type the endpoint for obtaining an authorization code.
- In the Token URI field, specify the endpoint used to exchange authentication credentials for access tokens.
- In the ServiceURL field, specify the endpoint for the Microsoft SharePoint service to which you are connecting.

OAuth 2.0 authentication

The driver supports OAuth 2.0 to access SharePoint resources. Before you can configure the driver for OAuth 2.0, you must register your application as an Azure Active Directory application and obtain the client information required for the OAuth flow or grant type you want to use for authentication.

The driver supports the following OAuth 2.0 flow and grant types for accessing web instances of SharePoint services.

- [Access token flow](#) on page 51
- [Refresh token grant](#) on page 51
- [Dynamic authorization code grant](#) on page 52

The following workflow describes the process for setting up OAuth 2.0 access.

1. [Register your application on the Azure portal](#): If it is not already registered, you need to register your application as an Azure Active Directory application to implement an OAuth 2.0 flow or grant and run the application against a SharePoint web instance.
2. [Obtain application client information and endpoints](#): To configure the driver for an OAuth 2.0 implementation, you need client credentials and endpoints related to your registered application.
3. [Obtain access and refresh tokens using the Configuration Manager](#): When using the access token flow or refresh token grant, you can obtain the access or refresh tokens using the Progress DataDirect Microsoft SharePoint Configuration Manager.
4. [Determine the scope](#): Scope refers to the permissions associated with an application. See this topic to know how to determine the scope for your application and specify it using the Scope connection option.
5. [Configure the driver to use OAuth 2.0](#): You can configure the driver to access SharePoint resources using any of the flow or grant types described in this topic.

Register your application on the Azure portal

Take the following steps to register your application as an Azure Active Directory application.

1. Go to the Azure portal.
<https://portal.azure.com>
2. Select **Azure Active Directory**.
3. Select **App registrations** on the left.
4. Select **New registration**.
5. Enter an application name.
6. Select the account types your application supports.
7. Enter a **Redirect URI** using one of the following methods:
 - If you are using the Configuration Manager to obtain OAuth tokens, select **Web** from the dropdown and enter `http://localhost` as the redirect URI.
 - If you are using a 3rd party application to obtain OAuth tokens, select **Web** or **Public client/native (mobile & desktop)** as required for your application. Then, enter the redirect URI associated with the application.
8. Press **Register**.
You are directed to the application overview page.
9. Add permissions associated with the Microsoft APIs your application uses.
 - a) Select **API permissions** from the navigation menu.
 - b) Select the API your application uses.
 - c) As appropriate, select **Delegated permissions** or **Application permissions**.

- d) Check the box of each permission you want to grant the application.
- e) Press **Add permissions**.

Results:

You have successfully registered your application on Azure Active Directory, as well as added permissions required for application access to Microsoft APIs.

Obtain application client information and endpoints

You must obtain the following client information and endpoints in order to retrieve the OAuth tokens necessary for connecting to a SharePoint site.

- Client ID - the client ID for your application
- Client Secret - the client secret for your application
- Authorization URI - the endpoint for obtaining an authorization code from the Azure authorization service
- Token URI - the endpoint used to exchange authentication credentials for access tokens

Take the following steps to obtain application client information and endpoints.

1. Go to the Azure portal.
<https://portal.azure.com>
2. Select **Azure Active Directory**.
3. Select **App registrations** on the left.
4. Select the app that you have created.
5. From the app overview page, record the ClientID.
6. Obtain your ClientSecret.
 - a) Select **Certificates & secrets**.
 - b) Click **New client secret**.
 - c) Enter description and choose an expiry option.
 - d) Click **Add**.
 - e) Copy the ClientSecret to a secure location.
7. Obtain authorization URI and token URI endpoints.
 - a) Return to the **Overview** page for your application.
 - b) Click **Endpoints** to display authorization and token endpoints associated with the application.

Two different APIs can be used to manage user and application permissions: Azure Active Directory (v1) and Microsoft Identity Platform (v2). As shown in the following examples, authorization and token endpoints have different formats, depending on which API you are using.

- Azure Active Directory (v1) authorization endpoint
`https://login.microsoftonline.com/common/oauth2/authorize`
- Microsoft Identity Platform (v2) authorization endpoint
`https://login.microsoftonline.com/common/oauth2/v2.0/authorize`

c) Record the AuthURI.

- For v1, the AuthURI consists of the **OAuth 2.0 authorization endpoint (v1)** appended with `?resource=<resource_uri>` where the `<resource_uri>` is the base URL for your SharePoint site. For example:

```
https://login.microsoftonline.com/common/oauth2/authorize?resource
=https://mycorp.sharepoint.com/
```

- For v2, the AuthURI is the **OAuth 2.0 authorization endpoint (v2)**. However, for v2, you must provide permissions via the Scope option. See [Determine the Scope](#) for details.

d) Record the TokenURI.

- For v1, the TokenURI is the **OAuth 2.0 token endpoint (v1)**.
- For v2, the TokenURI is the **OAuth 2.0 token endpoint (v2)**.

Obtain access and refresh tokens using the Configuration Manager

Note: The Configuration Manager is currently supported only on Windows platforms.

You need the following information before you begin.

- Service URL: the URL of the SharePoint site (or "site collection") to which you are connecting
- Authorization URI: the endpoint for obtaining an authorization code from the Azure authorization service
- Token URI: the endpoint used to exchange authentication credentials for access tokens
- Client ID: the client ID for your application
- Client Secret: the client secret for your application
- Scope: an OAuth scope or a space-separated list of OAuth scopes (only for Microsoft Identity Platform v2)

The following steps describe how you can use the Progress DataDirect Microsoft SharePoint Configuration Manager to obtain access and refresh tokens for either the access token flow or the refresh token grant. In addition, the Configuration Manager produces a connection URL that you can use in your application.

Note: You must allow popups in your browser to obtain access and refresh tokens with the Configuration Manager.

1. Open the Microsoft SharePoint Configuration Manager by selecting the ODBC Administrator from the Progress DataDirect program group.

For detailed information on launching the Configuration Manger, see "Configuring data sources with the Configuration Manger."

2. Set **Authentication Method** to 24-OAuth2.
3. Provide the following information in the fields provided.
 - **Service URL**
 - **Authorization URI**

- **Token URI**
- **Client ID**
- **Client Secret**
- **Scope** (only for Microsoft Identity Platform v2)

Note: If you are using the Microsoft Identity Platform (v2) to provision users and manage application access, the scope must include the `offline_access` permission to retrieve a refresh token. This permission allows the Configuration Manager to retrieve a refresh token, and, by way of the refresh token grant, provides the application with prolonged access to the SharePoint site. See also [Determine the Scope](#).

4. Retrieve access and refresh tokens.
 - a) Click **Fetch OAuth Token**.
 - b) If logon popup appears, enter Azure Active Directory credentials. (This popup may not appear if you previously logged on.)
 - c) If consent popup appears, provide consent, allowing the Configuration Manager to retrieve the tokens. (This popup may not appear if you previously provided consent to the Configuration Manager.)
 - d) The **Access Token** and **Refresh Token** fields populate with values retrieved from the OAuth authorization server.

5. Click **Test Connect** to verify connectivity and run SQL queries against the service.

Results:

The **Access Token** and **Refresh Token** fields include access and refresh tokens that you can use to implement OAuth 2.0.

The connection string in the **Connection String** field may be copied and used in your application to connect with your SharePoint service.

Note:

Not all the values in the resulting connection string are required. However, the connection string can be copied directly into your application. The driver ignores any values that do not apply to your OAuth implementation.

For example, the refresh token grant connection string, derived from the Configuration Manager, might include the following properties.

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;ServiceURL=serviceurl;  
AuthURI=auth_uri;TokenURI=token_uri;  
ClientID=client_id;ClientSecret=client_secret;  
AccessToken=access_token;RefreshToken=refresh_token;  
Scope=scope;
```

However, only the following properties are required for a refresh token grant connection string.

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;ServiceURL=serviceurl;  
TokenURI=token_uri;ClientID=client_id;  
ClientSecret=client_secret;RefreshToken=refresh_token;
```

Determine the scope

Scope refers to the permissions associated with an application. Scope is determined by your SharePoint administrator. If the Microsoft Identity Platform (v2) is being used to provision users and manage application access, then scope must be specified using the Scope option when retrieving the OAuth tokens necessary for connecting to a SharePoint site.

The value of the Scope option may consist of an OAuth scope or a space-separated list of OAuth scopes. The following syntax applies to each scope specified by the scope option.

```
resource_uri/scope_name offline_access [...]
```

where:

resource_uri

is the URI for your SharePoint site and is found at the start of the ServiceURL. For example, `https://mycorp.sharepoint.com` is the resource URI for the ServiceURL `https://mycorp.sharepoint.com/sites/marketing/global`.

scope_name

is the name of a scope being enforced against the SharePoint site.

offline_access

is a scope that enables prolonged access to resources on behalf of a user. This scope must be included if you are retrieving a refresh token.

Example

The following example shows the scope for a SharePoint site with the `.default` and `offline_access` scopes.

```
Scope=https://mycorp.sharepoint.com/.default offline_access
```

Note: The `.default` scope is a Microsoft Identity Platform scope that refers to the static list of permissions configured on the application registration. Refer to Microsoft Identity Platform documentation for further details.

Configure the driver to use OAuth 2.0

The driver supports the following OAuth 2.0 flow and grant types.

- [Access token flow](#) on page 51
- [Refresh token grant](#) on page 51
- [Dynamic authorization code grant](#) on page 52

See also

[Register your application on the Azure portal](#) on page 46

[Obtain application client information and endpoints](#) on page 47

[Obtain access and refresh tokens using the Configuration Manager](#) on page 48

[Connection option descriptions](#) on page 63

Access token flow

The access token flow passes the access token directly from the client to the SharePoint service for authentication.

Note: As opposed to using a third-party application such as Postman, you can use the Progress DataDirect Microsoft SharePoint Configuration Manager to obtain an access token to support the access token flow. See [Obtain access and refresh tokens using the Configuration Manager](#) for details.

Note: Access tokens are temporary and must be replaced to maintain the session without interruption. The life of an access token is typically one hour.

To configure the driver to use an access token flow:

- Set the ServiceURL property to the URL of the SharePoint site (or "site collection") to which you are connecting. For example, `https://mycorp.sharepoint.com/sites/marketing/global`.
- Set the AccessToken property to the value of the access token obtained from external sources.

The following examples show the connection information required to establish a session using the access token flow.

Connection URL

```
DRIVER=DataDirect 8.0 Microsoft Sharepoint;
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global/;
AccessToken=C3TQH9zjwek4CgJCU-4Mxb2DxLNfI2LB3a-dNfpWYx;
```

odbc.ini file

```
[Microsoft SharePoint]
Driver=ODBCHOME/lib/ivsharepoint28.so
...
Description=DataDirect 8.0 Microsoft SharePoint
...
AccessToken=C3TQH9zjwek4CgJCU-4Mxb2DxLNfI2LB3a-dNfpWYx
...
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global/
...
```

Refresh token grant

The refresh token grant is used to replace expired access tokens with active ones by exchanging the refresh token at the endpoint specified by the TokenURI option.

Note: As opposed to using a third-party application such as Postman, you can use the Progress DataDirect Microsoft SharePoint Configuration Manager to obtain a refresh token to support the refresh token grant. See [Obtain access and refresh tokens using the Configuration Manager](#) for details.

To configure the driver to use the refresh token grant:

- Set the ServiceURL property to the URL of the SharePoint site (or "site collection") to which you are connecting. For example, `https://mycorp.sharepoint.com/sites/marketing/global`.
- Set the ClientID property to specify the client ID key for your application.
- Set the ClientSecret property to specify the client secret for your application.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

- Set the TokenURI property to specify the endpoint from which the driver fetches access tokens.
- Set the RefreshToken property to specify the refresh token used to request a new access token or renew an expired one.

Important: The refresh token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

The following examples show the connection information required to establish a session using the refresh token grant.

Connection URL

```
DRIVER=DataDirect 8.0 Microsoft SharePoint;  
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global;  
ClientID=29453d6f-6gh-gd8g-44tk831;ClientSecret=12a3=bCD/Ef4Ik+Lm5P6s=/V+WylZcd;  
TokenURI=https://login.microsoftonline.com/common/oauth2/v2.0/token;  
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

odbc.ini file

```
[Microsoft SharePoint]  
Driver=ODBCHOME/lib/ivsharepoint28.so  
...  
Description=DataDirect 8.0 Microsoft SharePoint  
...  
ClientID=29453d6f-6789-25gh-gd8g-44tk3c527831  
...  
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXylZabcd  
...  
TokenURI=https://login.microsoftonline.com/common/oauth2/v2.0/token  
...  
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831  
...  
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global/  
...
```

Dynamic authorization code grant

Dynamic authorization code grant allows you to initiate an authorization code grant flow by specifying login credentials using the login prompt for your service, thereby providing a method to authenticate without fetching access and refresh tokens via the Configuration Manager or third-party application. Similar to authorization code grant, dynamic authorization code grant is typically used for web and native applications. It also provides secure connections by requiring multiple points of authentication before permitting access to data.

When connecting with dynamic authorization code grant flow, the driver launches the login prompt for your service in a separate browser window. After you submit your user and password credentials via the prompt, the driver exchanges your login credentials and client credentials for the Authorization Code from the location specified by the Authorization URI option. The driver then navigates to the endpoint specified by the Token URI option to exchange the authorization code for the access and refresh tokens. Finally, the application is redirected to the location provided in the Redirect URI option to begin the session.

After the grant flow is complete, the driver continues to use the access and refresh tokens to access data resources for the lifetime of the ODBC connection or until both the access and refresh tokens expire, whichever occurs first. If both tokens expire while the connection is still active, the driver launches the login prompt to reinitiate the flow.

Note: The dynamic authorization grant requires the manual submission of login credentials via the login prompt for your service; therefore, the driver does not support dynamic authorization grant in headless environments.

To use dynamic authorization code grant:

- Set the Authentication Method (`AuthenticationMethod`) option to 24 (OAuth2).
- Set the Enable Login Prompt (`EnableLoginPrompt`) option to 1 (enabled). When Enable Login Prompt is enabled, the driver launches the login prompt for your service in a separate browser window to initiate the OAuth grant flow.
- Set the SQL Engine Mode (`SQLEngineMode`) option to 2 (Direct). Note that this is the default setting for Windows.

Note: The dynamic authorization code grant is supported only in Direct mode.

- Set the Service URL (`ServiceURL`) option to the URL of the SharePoint site (or "site collection") to which you are connecting. For example, `https://mycorp.sharepoint.com/sites/marketing/global`.
- Set the Client ID (`ClientID`) option to specify the client ID key for your application.
- Set the Client Secret (`ClientSecret`) option to specify the client secret for your application.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

- Set the Authorization URI (`AuthURI`) option to specify the endpoint for obtaining an authorization code.
- Set the Token URI (`TokenURI`) option to specify the endpoint used to exchange authentication credentials for access tokens.
- Set the Scope (`Scope`) option to specify a space-separated list of OAuth scopes to limit the permissions granted by the access token.
- Set the Redirect URI (`RedirectURI`) option to specify the endpoint that the client is returned to after authenticating with a third-party service. Note that the value of the Redirect URI (`RedirectURI`) option must include the port number. For example, `RedirectURI=http://localhost:80` or `RedirectURI=http://localhost:8080`.
- Optionally, specify values for any additional options you want to configure. See "Connection option descriptions" for a complete list of options.

The following example demonstrates a basic session using the dynamic authorization code grant:

Connection URL:

```
Driver=DataDirect 8.0 Microsoft SharePoint;
AuthenticationMethod=24;EnableLoginPrompt=true;SQLEngineMode=2;
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global;
ClientID=abcdefghijklm3o4p5qr67s;ClientSecret=FaZBFRsGXTaR;
AuthURI=https://login.microsoftonline.com/xyzabc111000/oauth2/v2.0/authorize;
TokenURI=https://login.microsoftonline.com/xyz123abc/oauth2/token;
Scope=https://mycorp.sharepoint.com/offline_access;
RedirectURI=http://localhost:80;
```

odbc.ini File:

```
Driver=ODBCHOME/lib/ivsharepoint28.so
...
Description=DataDirect 8.0 Microsoft SharePoint
...
AuthenticationMethod=24
...
EnableLoginPrompt=true
...
SQLEngineMode=2
...
ServiceURL=https://mycorp.sharepoint.com/sites/marketing/global
...
ClientID=abcdefghijklm3o4p5qr67s
...
ClientSecret=FaZBFRsGXTaR
...
AuthURI=https://login.microsoftonline.com/xyzabc111000/oauth2/v2.0/authorize
...
TokenURI=https://login.microsoftonline.com/xyz123abc/oauth2/token
...
Scope=https://mycorp.sharepoint.com/offline_access
...
RedirectURI=http://localhost:80
...
```

See also

[Connection option descriptions](#) on page 63

Performance considerations

Application Using Threads (ApplicationUsingThreads): The driver coordinates concurrent database operations (operations from different threads) by acquiring locks. Although locking prevents errors in the driver, it also decreases performance. If your application does not make ODBC calls from different threads, the driver has no reason to coordinate operations. In this case, the ApplicationUsingThreads attribute should be disabled (set to 0).

Note: If you are using a multi-threaded application, you must enable the Application Using Threads option.

Batch Update Chunk Size (BatchUpdateChunkSize): This option can impact performance by reducing the number of round trips required to complete an insert, update, or delete operation. For example, at the default setting of 200 rows, a 10,000 row insert operation would require 50 round trips. Setting Batch Update Chunk Size to 500 rows would reduce the number of round trips to 20.

Create Map (CreateMap): Create Map determines whether the driver creates the internal files required for a relational map of the native data when establishing a connection. When the driver creates these internal files, an initial connection can take a few minutes, depending on network speeds and the amount of metadata the driver must retrieve from the service.

Fetch Size (FetchSize): Reducing the number of round trips on the network to the approximate number of rows being fetched increases performance. For example, if your application normally fetches 200 rows, it is more efficient for the driver to fetch 200 rows at one time over the network than to fetch 50 rows at a time during four round trips over the network.

JVM Arguments (JVMArgs): This connection option can be used to address memory and performance concerns by adjusting the max Java heap size. By increasing the max Java heap size, you increase the amount of data the driver accumulates in memory. This can reduce the likelihood of out-of-memory errors and improve performance by ensuring that result sets fit easily within the JVM's free heap space.

Web Service Compress Data (WSCCompressData): Web Service Compress Data allows you to compress data sent to and from the Web server. By enabling this connection option, you can significantly improve performance by reducing the size of data transferred between the driver and the Web server.

See also

[Connection option descriptions](#) on page 63

Using the SQL engine server

Some applications may experience problems loading the JVM required for the SQL engine because the process exceeds the available heap space. If your application experiences problems loading the JVM, you can configure the driver to operate in server mode.

The driver supports the following SQL engine modes:

- **Server mode:** The driver's SQL engine runs in a separate process with its own JVM, instead of trying to load the SQL engine and JVM in the same process used by the driver.
- **Direct mode:** The driver operates with the SQL engine and JVM running in a single process.
- **Auto mode:** The driver attempts to run in server mode. However, if server mode is unavailable, the SQL engine will failover to run in direct mode.

By default:

- **Windows:** The driver operates in server mode by default.
- **Linux:** The driver operates in direct mode by default.

Note: You must be an administrator to start or stop the service, or to configure any settings for the service.

See the following sections for details on configuring the SQL engine server on your platform.

For details, see the following topics:

- [Configuring server mode using the Configuration Manager](#)
- [Configuring the SQL engine server using Java options](#)
- [Configuring Java logging for the SQL engine server](#)

Configuring server mode using the Configuration Manager

In server mode, you must start the SQL engine server before connecting.

Note: The Configuration Manager is currently supported only on Windows platforms.

The following sections describe how to configure, start, and stop the SQL engine server using the Configuration Manager.

1. Open your data source using the Configuration Manager; then, select the **SQL Engine** tab.
2. Set the SQL Engine Mode (SQLEngineMode) connection option to one of the following values:
 - **0 - Auto:** The SQL engine attempts to run in server mode first, but will failover to direct mode if server mode is unavailable.
 - **1 - Server:** The SQL engine runs exclusively in server mode.

Note: By default, SQL Engine Mode is set to **1 - Server** for Windows and **2 - Direct** for Linux.

The fields associated with server mode will become editable, and the **Start** button appears.

3. Provide values for the following options:
 - **Server Port Number:** Specifies a valid port on which the SQL engine listens for requests from the driver. The default value depends on your platform:
 - 32-bit: 19964
 - 64-bit: 19963
 - **JVM Classpath:** Specifies the CLASSPATH for the JVM used by the driver. See "JVM Classpath" for details. The default depends on your platform:
 - Windows: {.;c:\install_dir\java\lib\sharepoint.jar}
 - Linux: {./home/user1/install_dir/java/lib/sharepoint.jar}
 - **JVM Arguments:** A string that contains the arguments that are passed to the JVM that the driver is starting. The location of the JVM must be specified on your PATH. See "JVM Arguments" for details. The default value is:
 - Xmx1024m
 - **JVM Path:** Specifies fully qualified path to the JVM executable that you want to use to run the SQL engine server. The path must not contain double quotation marks.
4. Optionally, if connecting through a proxy server, provide values for the following options:
 - **Server Proxy Host:** Specifies the host name of the proxy server used by the SQL engine. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.
 - **Server Proxy Port:** Specifies the port needed to connect to the proxy server used by the SQL engine.
 - **Server Proxy User:** Specifies the user name needed to connect to the proxy server used by the SQL engine.

- **Server Proxy Password:** Specifies the password needed to connect to the proxy server used by the SQL engine.

Note: After the initial configuration, in order for changes to the required and optional connection option values to take effect, you must restart the SQL engine server.

5. Click **Save** to save your changes
6. Click **Start** to run the SQL engine service. A message is displayed that indicates whether the SQL engine was able to successfully run.

See also

[SQL Engine Mode](#) on page 93

[Server Port Number](#) on page 88

[JVM Classpath](#) on page 79

[JVM Arguments](#) on page 78

[JVM Path](#) on page 80

[Server Proxy Host](#) on page 89

[Server Proxy Port](#) on page 90

[Server Proxy User](#) on page 91

[Server Proxy Password](#) on page 90

Stopping the SQL engine server using the Configuration Manager

To stop the SQL engine server:

1. Open the Configuration Manager and select the SQL Engine tab.
2. From the SQL Engine Mode drop-down list, select **0 - Auto** or **1 - Server**.
3. Click **Stop** to stop the service. A message is displayed to confirm that the service stopped.
4. Click **Exit** to close the Configuration Manager.

Configuring the SQL engine server using Java options

Before you begin: In server mode, you must start the SQL engine server before using the driver.

The following sections describe how to configure, start, and stop the SQL engine server on Linux platform.

Note: By default, SQL Engine Mode is set to 1 (Server) for Windows and 2 (Direct) for Linux.

To configure the SQL engine server, specify values for the Java options in the following JVM argument to suit your environment. See the "SQL engine server Java options" table for a description of these options.

```
java -Xmx<heap_size>m -cp "<jvm_classpath>" com.ddtek.jdbc.<driver>.phoenix.sql.Server
    -port <port_number> -Dhttp.proxyHost=<proxy_host> -Dhttp.proxyPort=<proxy_port>
    -Dhttp.proxyUser=<proxy_user> -Dhttp.proxyPassword=<proxy_password>
```

For example:

```
java -Xmx1024m -cp "/opt/Progress/DataDirect/ODBC_80_64bit/java/lib/sharepoint.jar"
com.ddtek.phoenix.sql.Server -port 19963 -Dhttp.proxyHost=myhost@mydomain.com
-Dhttp.proxyPort=12345 -Dhttp.proxyUser=JohnQPublic -Dhttp.proxyPassword=secret
```

To start the SQL engine service, execute the JVM Argument after configuring the Java options. A confirmation message is returned once the server is online.

Note: After the initial configuration, in order for changes to these connection option values to take effect, you must restart the SQL engine server.

Table 2: SQL engine server Java options

Java option	Description
Required Java options	
-cp	Specifies the CLASSPATH for the Java Virtual Machine (JVM) used by the driver. The CLASSPATH is the search string the JVM uses to locate the Java jar files the driver needs. The driver's JVM is located on the following path: <i>install_dir/java/lib/sharepoint.jar</i>
-port	Specifies a valid port on which the SQL engine listens for requests from the driver. We recommend specifying one of the following values: <ul style="list-style-type: none"> • 19964 (32-bit drivers) • 19963 (64-bit drivers)
Optional Java options	
-Xmx	Specifies the maximum memory heap size, in megabytes, for the JVM. The default size is determined by your JVM. We recommend specifying a size no smaller than 1024. Note: Although this option is not required to start the SQL engine server, we highly recommend specifying a value.
-Dhttp.proxyHost	Specifies the host name of the proxy server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.
-Dhttp.proxyPort	Specifies the port number where the proxy server is listening for HTTP and/or HTTPS requests.
-Dhttp.proxyUser	Specifies the user name needed to connect to the proxy server.
-Dhttp.proxyPassword	Specifies the password needed to connect to the proxy server.

Stopping the SQL engine server

To stop the SQL engine server, choose one of the following:

- Using an application, execute `SHUTDOWN SQL`.
- From a command line, press `Ctrl + C`.

A message is returned to confirm that the service is stopped.

Configuring Java logging for the SQL engine server

Java logging for the SQL engine server can be configured using either the JVM or the driver.

For details, refer to "Configuring logging" in the *Progress DataDirect for ODBC Drivers Reference*.

Connection option descriptions

The connection option descriptions in this section are listed alphabetically by the GUI name that appears on the driver Setup dialog box. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name in the option topics.

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the following tables of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions by their attribute names.

Note: The driver does not support specifying values for the same connection option multiple times in a connection string or DSN. If a value is specified using the same attribute multiple times or using both long and short attributes, the connection may fail or the driver may not behave as intended.

The following tables provide a summary of supported connection options by functionality, including their attribute names, short names, and default values.

[General options](#)

[OAuth 2.0 authentication options](#)

[Mapping options](#)

[SQL Engine options](#)

[Proxy server options](#)

[Web service options](#)

[Additional options](#)

General options

The following table summarizes general connection options that can apply to all connections that use data sources.

Table 3: General options

Attribute (Short Name)	Default
DataSourceName (dsn)	No default value
Description (desc)	No default value
ServiceURL (surl)	No default value

OAuth 2.0 authentication options

The following table summarizes options used for OAuth 2.0 authentication method.

Table 4: OAuth 2.0 authentication options

Attribute (Short Name)	Default
AccessToken (atok)	No default value
AuthenticationMethod (am)	24 (OAuth2)
AuthURI (o2au)	No default value
ClientID (cid)	No default value
ClientSecret (clse)	No default value
EnableLoginPrompt (elp)	0 (false)
RedirectURI (o2ru)	No default value
RefreshToken (rtok)	No default value
Scope (oas)	No default value
ServiceURL (surl)	No default value
TokenURI (o2tu)	No default value

Mapping options

The following table summarizes connection properties involved in mapping the Microsoft SharePoint data model to a SQL model.

Table 5: Mapping options

Attribute (Short Name)	Default
AuditColumns (audc)	0 (None)

Attribute (Short Name)	Default
CreateMap (cm)	0 (NotExist)
KeywordConflictSuffix (kcs)	No default value
RefreshSchema (rs)	0 (False)
SchemaMap (smp)	Default value depends on environment

SQL engine options

The following table lists the options used to configure the SQL engine.

Table 6: SQL Engine options

Attribute (Short Name)	Default
JVMArgs (jvma)	For the 32-bit driver when the SQL Engine Mode is set to 2 (Direct): -Xmx256m For all other configurations: -Xmx1024m
JVMClassPath (jvmcp)	No default value
JVMPath (jvmp)	<i>install_dir\jre\bin\java.exe</i>
ServerPortNumber (sport)	For 32-bit: 19964 For 64-bit: 19963
ServerProxyHost (sph)	No default value
ServerProxyPassword (spw)	No default value
ServerProxyPort (spp)	No default value
ServerProxyUser (spu)	No default value
SQLEngineMode (sem)	For Windows: 1 (Server) For Linux: 2 (Direct)
SQLService (ss)	No default value

Proxy server options

The following table summarizes proxy server connection options.

Table 7: Proxy server options

Attribute (Short Name)	Default
ProxyHost (pxhn)	No default value
ProxyPassword (pxpw)	No default value
ProxyPort (pxpt)	0 which means that the default value is determined by whether the value specified for the Proxy Host (ProxyHost) option is an HTTP or HTTPS URL. For HTTP: 80 For HTTPS: 443
ProxyUser (pxun)	No default value

Web service options

The following table summarizes Web service connection options, including those related to timeouts.

Table 8: Web service options

Attribute (Short Name)	Default
StmtCallLimit (scl)	0 (no limit)
StmtCallLimitBehavior (sclb)	1 (ErrorAlways)
WSCompressData (wscd)	0 (Compress)

Additional options

The following table summarizes additional connection options.

Table 9: Additional Options

Attribute (Short Name)	Default
ApplicationUsingThreads (aut)	1 (True)
BatchUpdateChunkSize (bucs)	200 (rows)
DateTimeZoneShift (dtzs)	0 (minutes)
ExtendedOptions (xo)	No default value
FetchSize (fs)	100 (rows)
InitializationString (is)	No default value
LogConfigFile (lgcf)	ddlogging.properties

Attribute (Short Name)	Default
ReadOnly (ro)	0 (false)
ReportCodepageConversionErrors (rcce)	0 (Ignore Errors)
TransactionMode (tm)	0 (NoTransactions)

For details, see the following topics:

- [Access Token](#)
- [Application Using Threads](#)
- [Audit Columns](#)
- [Authentication Method](#)
- [Authorization URI](#)
- [Batch Update Chunk Size](#)
- [Client ID](#)
- [Client Secret](#)
- [Create Map](#)
- [Data Source Name](#)
- [DateTime Zone Shift](#)
- [Description](#)
- [Enable Login Prompt](#)
- [Extended Options](#)
- [Fetch Size](#)
- [Initialization String](#)
- [JVM Arguments](#)
- [JVM Classpath](#)
- [JVM Path](#)
- [Keyword Conflict Suffix](#)
- [Log Config File](#)
- [Proxy Host](#)
- [Proxy Password](#)
- [Proxy Port](#)
- [Proxy User](#)
- [Read Only](#)

- [Redirect URI](#)
- [Refresh Schema](#)
- [Refresh Token](#)
- [Report Codepage Conversion Errors](#)
- [Schema Map](#)
- [Scope](#)
- [Server Port Number](#)
- [Server Proxy Host](#)
- [Server Proxy Password](#)
- [Server Proxy Port](#)
- [Server Proxy User](#)
- [Service URL](#)
- [Statement Call Limit](#)
- [Statement Call Limit Behavior](#)
- [SQL Engine Mode](#)
- [SQL Service](#)
- [Token URI](#)
- [Transaction Mode](#)
- [Web Service Compress Data](#)

Access Token

Attribute

AccessToken (atok)

Purpose

Specifies the access token used to authenticate to Microsoft SharePoint with OAuth 2.0 enabled. Typically, this option is configured by the application; however, in some scenarios, you may need to secure a token using external processes. In those instances, you can also use this option to set the access token manually.

Valid Values

String

where:

String

is an access token you have obtained from the authentication service.

Notes

- Access tokens are temporary and must be replaced to maintain the session without interruption. The life of an access token is typically one hour.
- See "OAuth 2.0 authentication" for examples and more information.

Default Value

No default value

See also

[OAuth 2.0 authentication](#) on page 45

Application Using Threads

Attribute

ApplicationUsingThreads (aut)

Purpose

Determines whether the driver works with applications using multiple ODBC threads.

Valid Values

0 | 1

Behavior

If set to 1 (true), the driver works with single-threaded and multi-threaded applications.

If set to 0 (false), the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Notes

- This connection option can affect performance.

Default Value

1 (true)

See also

[Performance considerations](#) on page 54

Audit Columns

Attribute

AuditColumns (audc)

Purpose

Determines whether the driver includes Microsoft SharePoint internal columns (or *audit columns*) when mapping the Microsoft SharePoint data model to the relational data model.

Valid Values

0 | 1

Behavior

If set to 0 (None), the driver does not include internal columns in the relational data model.

If set to 1 (All), the driver includes internal columns in the relational data model.

Default Value

0 (None)

Authentication Method

Attribute

AuthenticationMethod (am)

Purpose

Determines which authentication method the driver uses during the course of a session. OAuth 2.0 is currently the only supported authentication method. Therefore, AuthenticationMethod does not need to be specified. However, it does appear on the Configuration Manager's **Connection** tab.

Valid Values

24

Behavior

If set to 24 (OAuth2), the driver uses OAuth 2.0 to authenticate to endpoints. See "OAuth 2.0 authentication" for details.

Default Value

24 (OAuth2)

See also

[OAuth 2.0 authentication](#) on page 45

Authorization URI

Attribute

AuthURI (o2au)

Purpose

Specifies the endpoint for obtaining an authorization code from a third-party authorization service for OAuth 2.0 implementations.

Valid Values

String

where:

String

is the endpoint for retrieving the OAuth 2.0 authorization code from the third party authorization service.

Notes

- When this endpoint is queried, the authorization service presents an interface prompting the user to approve or deny access to backend data.
- See "OAuth 2.0 authentication" for examples and more information.

Default Value

No default value

See also

[OAuth 2.0 authentication](#) on page 45

Batch Update Chunk Size

Attribute

BatchUpdateChunkSize (bucs)

Purpose

Specifies the maximum number of rows sent to the service when an insert, update, or delete operation impacts multiple rows.

Valid Values

x

where:

x

is a number of rows greater than or equal to 1.

Notes

This option can impact performance by reducing the number of round trips required to complete an insert, update, or delete operation. For example, at the default setting of 200 rows, a 10,000 row insert operation would require 50 round trips. Setting Batch Update Chunk Size to 500 rows would reduce the number of round trips to 20.

Default Value

200

See also

[Performance considerations](#) on page 54

Client ID

Attribute

ClientID (cid)

Purpose

Specifies the client ID key for your application when authenticating to Microsoft SharePoint with OAuth 2.0 enabled.

Valid Values

String

where:

String

is the client ID key for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

[Performance considerations](#) on page 54

Client Secret

Attribute

ClientSecret (clse)

Purpose

Specifies the client secret for your application when authenticating to Microsoft SharePoint with OAuth 2.0 enabled.

Important: The client secret is a confidential value used to authenticate the application to the service. To prevent unauthorized access, this value must be securely maintained.

Valid Values

String

where:

String

is the client secret for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

[Performance considerations](#) on page 54

Create Map

Attribute

CreateMap (cm)

Purpose

Determines whether the driver creates the internal files required for a relational map of the native data model when establishing a connection.

Valid Values

0 | 1 | 2 | 3

Behavior

If set to 0 (NotExist), the driver uses the current group of internal files specified by the Schema Map (SchemaMap) option. If the files do not exist, the driver creates them.

If set to 1 (ForceNew), the driver deletes the group of internal files specified by the Schema Map option and creates a new group of these files at the same location.

Warning: 1 (ForceNew) causes all views, data caches, and map customizations defined in the current schema map to be lost.

If set to 2 (No), the driver uses the current group of internal files specified by the Schema Map option. If the files do not exist, the connection fails.

If set to 3 (Session), the driver uses memory to store the internal configuration information and relational map of the native data model. After the session, the view is discarded.

Notes

- The internal files share the same directory as the schema map configuration file. This directory is specified with the Schema Map connection option.

- You can refresh the internal files related to an existing relational view of your data with the SQL extension Refresh Map. Refresh Map runs a discovery against your native data and updates your internal files accordingly.

Default Value

3 (Session)

See also

[Performance considerations](#) on page 54

Data Source Name

Attribute

DataSourceName (dsn)

Purpose

Specifies the name of a data source in your Windows Registry or `odbc.ini` file.

Valid Values

String

where:

String

is the name of a data source.

Default Value

No default value

DateTime Zone Shift

Attribute

DateTimeZoneShift (dtzs)

Purpose

Specifies the time zone shift in minutes to DateTime values. If no value is specified, no time zone shift is applied. The application of a time zone shift may be necessary to synchronize client DateTime values with those on the server.

Valid Values

x

where:

x

is a number of minutes between -1440 and 1440.

Behavior

If set to *x*, the number of minutes specified is added to DateTime values.

If set to *-x*, the number of minutes specified is subtracted from DateTime values.

Default Value

0

Description

Attribute

Description (desc)

Purpose

Specifies an optional long description of a data source. This description is not used as a runtime connection attribute, but does appear in the `ODBC.INI` section of the Registry and in the `odbc.ini` file.

Valid Values

String

where:

String

is a description of a data source.

Default Value

No default value

Enable Login Prompt

Attribute

EnableLoginPrompt (elp)

Purpose

Specifies whether the driver fetches access and refresh tokens at connection when OAuth 2.0 Authorization Code Grant is enabled (`AuthenticationMethod=51`). When this option is enabled, the driver launches the login prompt for your service at connection, which allows you to specify your login credentials and initiate the dynamic authorization code grant flow. Enabling this option provides a method of fetching access and refresh tokens without using the Configuration Manager or third-party tool.

Valid Values

0 | 1

Behavior

If set to 0 (false), the driver does not launch the login prompt for your service. If access and refresh tokens are needed for your authentication flow, you will need to fetch them using the Configuration Manager or another tool.

If set to 1 (true), the driver opens the login prompt for your service when attempting to connect. Submitting user and password credentials via the prompt initiates the dynamic authorization code grant to fetch access and refresh tokens.

Notes

- This option is used only for the dynamic authorization code grant flow. See "Dynamic authorization code grant" for a full list of requirements.
- This option is supported only when the SQL Engine is running in direct mode (`SQLEngineMode=2`).
- When this option is enabled, the value of the Redirect URI (`RedirectURI`) option must include the port number. For example, `RedirectURI=http://localhost:80` or `RedirectURI=http://localhost:8080`.

Default Value

0 (false)

See also

[Dynamic authorization code grant](#) on page 52

Extended Options

Attribute

ExtendedOptions (xo)

Purpose

Specifies a semicolon separated list of connection options and their values. Use this connection option to set the value of undocumented connection options that are provided by Progress DataDirect Technical Support.

Valid Values

option=value[;option=value;...]

where:

option

is a connection option.

value

is the value or setting of the connection option.

Default Value

No default value

Fetch Size

Attribute

FetchSize (fs)

Purpose

Specifies the maximum number of rows that the driver processes before returning data to the application when executing a Select. This value provides a suggestion to the driver as to the number of rows it should internally process before returning control to the application. The driver may fetch fewer rows to conserve memory when processing exceptionally wide rows.

Valid Values

0 | x

where:

x

is a positive integer indicating the number of rows that should be processed.

Behavior

If set to 0, the driver processes all the rows of the result before returning control to the application. When large data sets are being processed, setting Fetch Size to 0 can diminish performance and increase the likelihood of out-of-memory errors.

If set to x , the driver limits the number of rows that may be processed for each fetch request before returning control to the application.

Notes

- To optimize throughput and conserve memory, the driver uses an internal algorithm to determine how many rows should be processed based on the width of rows in the result set. Therefore, the driver may process fewer rows than specified by Fetch Size when the result set contains exceptionally wide rows. Alternatively, the driver processes the number of rows specified by Fetch Size when the result set contains rows of unexceptional width.
- Fetch Size can be used to adjust the trade-off between throughput and response time. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes can improve overall response times at the cost of additional memory.
- You can use Fetch Size to reduce demands on memory and decrease the likelihood of out-of-memory errors. Simply, decrease Fetch Size to reduce the number of rows the driver is required to process before returning data to the application.

Default Value

100

See also

[Performance considerations](#) on page 54

Initialization String

Attribute

InitializationString (is)

Purpose

Specifies one or multiple SQL commands to be executed by the driver after it has established a connection and has performed all initialization for the connection. If the execution of a SQL command fails, the connection attempt also fails and the driver throws an exception indicating which SQL command or commands failed.

Valid Values

command[[;*command*]...]

where:

command

is a SQL command.

Notes

Multiple commands must be separated by semicolons. In addition, if this option is specified in a connection URL, the entire value must be enclosed in parentheses when multiple commands are specified.

Default Value

No default value

JVM Arguments

Attribute

JVMArgs (jvma)

Purpose

A string that contains the arguments that are passed to the JVM that the driver is starting. The location of the JVM must be specified on the driver library path. For information on setting the location of the JVM in your environment, see:

- "Windows environment variables"
- "Library search path" (Linux)

When specifying the heap size for the JVM, the JVM tries to allocate the heap memory as a single contiguous range of addresses in the application's memory address space. If the application's address space is fragmented so that there is no contiguous range of addresses big enough for the amount of memory specified for the JVM, the driver fails to load, because the JVM cannot allocate its heap. This situation is typically encountered only with 32-bit applications, which have a much smaller application address space. If you encounter problems with loading the driver in an application, try reducing the amount of memory requested for the JVM heap. If possible, switch to a 64-bit version of the application.

Valid Values

String

where:

String

contains arguments that are defined by the JVM. Values that include special characters or spaces must be enclosed in curly braces { } when used in a connection string.

Example

To set the heap size used by the JVM to 256 MB and the http proxy information, specify:

```
{-Xmx256m -Dhttp.proxyHost=johndoe -Dhttp.proxyPort=808}
```

To set the heap size to 256 MB and configure the JVM for remote debugging, specify:

```
{-Xmx256m -Xrunjdwp:transport=dt_socket, address=9003,server=y,suspend=n -Xdebug}
```

Default Value

For the 32-bit driver when the SQL Engine Mode connection option is set to 2 (Direct):

```
-Xmx256m
```

For all other configurations:

```
-Xmx1024m
```

See also

[Windows environment variables](#) on page 32

[Library search path](#) on page 32

JVM Classpath

Attribute

JVMClassPath (jvmcp)

Purpose

Specifies the CLASSPATH for the Java Virtual Machine (JVM) used by the driver. The CLASSPATH is the search string the JVM uses to locate the Java jar files the driver needs.

Valid Values

string

where:

string

specifies the CLASSPATH. Separate multiple jar files by a semi-colon on Windows platforms and by a colon on Linux platforms. CLASSPATH values with multiple jar files must be enclosed in curly braces { } when used in a connection string.

If your process employs multiple drivers that use a JVM, the value of the JVM Classpath for all affected drivers must include an absolute path to all the jar files for drivers used in that process. In addition, the value specified must be identical for all drivers. For example if you are using the SharePoint and MongoDB drivers on Windows, you would specify a value of {c:\install_dir\java\lib\sharepoint.jar; c:\install_dir\java\lib\mongodb.jar} for both drivers. If the value for any of the affected drivers is missing a file path or is different from the one specified for the other drivers, the drivers will return an error at connection that the JVM is already running.

Example

On Windows:

```
{.;c:\install_dir\java\lib\sharepoint.jar}
```

On Linux:

```
{./home/user1/install_dir/java/lib/sharepoint.jar}
```

Default Value

No default value

JVM Path

Attribute

JVMPath (jvmp)

Purpose

Specifies fully qualified path to the JVM executable that you want to use to run the SQL Engine Server. The path must not contain double quotation marks.

Valid Values

String

where:

String

The full path to the JVM executable.

Default Value

install_dir\jre\bin\java.exe

See also

[Using the SQL engine server](#) on page 57

Keyword Conflict Suffix

Attribute

KeywordConflictSuffix (kcs)

Purpose

Specifies a string of up to 5 alphanumeric characters that the driver appends to any object or field name that conflicts with a SQL engine keyword.

Valid Values

String

where:

String

is a string of up to 5 alphanumeric characters.

Example

A field called `CASE` exists in the data schema. To avoid a naming conflict with the SQL engine keyword `CASE`, you could set `KeywordConflictSuffix=TAB`. In this scenario, the driver maps the `CASE` field to the `CASETAB` column.

Default Value

No default value

Log Config File

Attribute

LogConfigFile (lgcf)

Purpose

Specifies the file name, and optionally, the path of the properties file used to initialize driver logging.

Valid Values

String

where:

String

is the relative or fully qualified path of the properties file to load to initialize driver logging. If you do not specify a path, the driver looks for this file in the current working directory. If the specified file does not exist, the driver continues searching for an appropriate properties file as described in "Logging for Java components" in the *Progress DataDirect for ODBC Drivers Reference*.

Default Value

`ddlogging.properties`

Proxy Host

Attribute

ProxyHost (pxhn)

Purpose

Identifies a proxy server to use for the first connection.

Valid Values

server_name | *IP_address*

where:

server_name

is the name of the proxy server, which may be qualified with the domain name.

IP_address

is an IP address, specified in either IPv4 or IPv6 format, or a combination of the two.

Default Value

No default value

Proxy Password

Attribute

ProxyPassword (pxpw)

Purpose

Specifies the password needed to connect to a proxy server for the first connection.

Valid Values

password

where:

password

is a valid password for that server. Contact your system administrator to obtain a valid password.

Default Value

No default value

Proxy Port

Attribute

ProxyPort (pxpt)

Purpose

Specifies the port number where the proxy server is listening for HTTP or HTTPS requests for the first connection.

Valid Values

port

where:

port

is the port number on which the proxy server is listening. Contact your system administrator to obtain the correct port.

Default Value

0 which means that the default value is determined by whether the value specified for the Proxy Host (ProxyHost) option is an HTTP or HTTPS URL.

For HTTP: 80

For HTTPS: 443

See also

[Proxy Host](#) on page 82

Proxy User

Attribute

ProxyUser (pxun)

Purpose

Specifies the user name needed to connect to a proxy server for the first connection.

Valid Values

user_name

where:

user_name

is a valid user ID for the proxy server.

Default Value

No default value

Read Only

Attribute

ReadOnly (ro)

Purpose

Specifies whether the connection has read-only access to the data source.

Valid Values

0 | 1

Behavior

If set to 1 (true), the connection has read-only access.

If set to 0 (false), the connection is opened for read/write access, and you can use all commands supported by the product.

Default Value

1 (true)

Redirect URI

Attribute

RedirectURI (o2ru)

Purpose

Specifies the endpoint to which the client is returned after third-party authorization for OAuth 2.0 implementations.

Valid Values

String

where:

String

is the endpoint to which the client is returned after third-party authorization. For example, `http://localhost`.

Notes

- The redirect URI is often registered with the authentication service to provide improved security. Registering the endpoint prevents your valid authentication credentials being redirected to a malicious site; therefore, reducing the risk of sharing your access token and other sensitive information with unauthorized parties.

- See "OAuth 2.0 authentication" for examples and more information.

Default Value

No default value

See also

[OAuth 2.0 authentication](#) on page 45

Refresh Schema

Attribute

RefreshSchema (rs)

Purpose

Specifies whether the driver automatically refreshes the relational map of the schema when a user connects to the service.

Valid Values

0 | 1

Behavior

If set to 1 (true), the driver automatically refreshes the map when a user first connects to the service. The driver rebuilds the relational map of the schema, and any changes that have been made to the schema since the last refresh will be shown in the metadata.

If set to 0 (false), the driver does not refresh the relational map when a user first connects.

Notes

- This option should not be enabled (`RefreshSchema=1`) when `CreateMap=3` (Session).
- This option is equivalent to executing the Refresh Map statement.

Default Value

0 (false)

Refresh Token

Attribute

RefreshToken (rtok)

Purpose

Specifies the refresh token used to either request a new access token or renew an expired access token for OAuth 2.0 implementations.

Important: The refresh token is a confidential value used to authenticate to the service. To prevent unauthorized access, this value must be securely maintained.

Valid Values

String

where:

String

is the refresh token you have obtained from the authentication service.

Notes

- See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

[OAuth 2.0 authentication](#) on page 45

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors (rcce)

Purpose

Specifies how the driver handles code page conversion errors that occur when a character cannot be converted from one character set to another.

An error message or warning can occur if an ODBC call causes a conversion error, or if an error occurs during code page conversions to and from the service or to and from the application. The error or warning generated is `Code page conversion error encountered`. In the case of parameter data conversion errors, the driver adds the following sentence: `Error in parameter x`, where `x` is the parameter number. The standard rules for returning specific row and column errors for bulk operations apply.

Valid Values

0 | 1 | 2

Behavior

If set to 0 (IgnoreErrors), the driver substitutes 0x1A for each character that cannot be converted and does not return a warning or error.

If set to 1 (ReturnError), the driver returns an error instead of substituting 0x1A for unconverted characters.

If set to 2 (ReturnWarning), the driver substitutes 0x1A for each character that cannot be converted and returns a warning.

Default Value

0

Schema Map

Attribute

SchemaMap (smp)

Purpose

Specifies either the name or the absolute path and name of the configuration file where the map of the database data model is written. The driver looks for this file when connecting to a database instance. If the file does not exist, the driver creates one.

Valid Values

string

where:

string

is either the name or the absolute path and name (including the `.config` extension) of the configuration file. For example, if specifying a value of:

- ABC, the driver either creates or looks for the configuration file ABC in the working directory of your application.
- C:\\Users\\Default\\AppData\\Local\\Progress\\DataDirect\\<driver>_Schema\\abc@defcorp.com.config, the driver either creates or looks for the configuration file abc@defcorp.com.config in the directory C:\\Users\\Default\\AppData\\Local\\Progress\\DataDirect\\<driver>_Schema.

Notes

- When connecting to a database instance, the driver looks for the schema map configuration file. If the configuration file does not exist, the driver creates the schema map configuration file using the name and location you have provided. If you do not provide a name and location for the configuration file, the driver creates it using default values.
- The driver uses the path specified in this connection option to store additional internal files.
- You can refresh the internal files related to an existing view of your data by using the SQL extension Refresh Map. Refresh Map runs a discovery against your native data and updates your internal files accordingly.

Default Value

The default is determined by the environment. The driver attempts to create the files in a subdirectory of the first available directory in the following order:

- Windows
 - DD_HOME environment variable
 - LOCALAPPDATA environment variable

- APPDATA environment variable
- User's home directory
- For UNIX/Linux
 - DD_HOME environment variable
 - User's home directory

For both Windows and UNIX/Linux, the file path takes the following format:

```
<available_location>/progress/datadirect/<driver>_schema/<user_name>.config
```

Scope

Attribute

Scope (oas)

Purpose

Specifies a space-separated list of OAuth scopes that limit the permissions granted by an access token.

Valid Values

String

where:

String

is a space-separated list of security scopes.

Default Value

No default value

Server Port Number

Attribute

ServerPortNumber (sport)

Purpose

Specifies a valid port on which the SQL engine listens for requests from the driver.

Valid Values

port_number

where:

port_number

is the port number of the server listener. Check with your system administrator for the correct number.

Notes

- This option is ignored when SQL Engine Mode (SQLEngineMode) is set to 2 (Direct).

Default Value

32-bit: 19964

64-bit: 19963

See also

[Using the SQL engine server](#) on page 57

Server Proxy Host

Attribute

ServerProxyHost (sph)

Purpose

Specifies the host name and possibly the domain of the proxy server used by the SQL engine server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.

Valid Values

server_name | *ip_address*

where:

server_name

is the name of the proxy server or a fully qualified domain name to which you want to connect.

IP_address

is the IP address of the server. The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two.

Default Value

No default value

See also

[Using the SQL engine server](#) on page 57

Server Proxy Password

Attribute

ServerProxyPassword (spw)

Purpose

Specifies the password needed to connect to the proxy server used by the SQL engine server.

Valid Values

string

where:

string

specifies the password to use to connect to the Proxy Server. Contact your system administrator to obtain your password.

Default Value

No default value

See also

[Using the SQL engine server](#) on page 57

Server Proxy Port

Attribute

ServerProxyPort (spp)

Purpose

Specifies the port number of the server listener for the proxy server used by the SQL engine server.

Valid Values

port_name

where:

port_name

is the port number of the server listener of the proxy server used by the SQL engine server. Check with your system administrator for the correct number.

Default Value

No default value

See also

[Using the SQL engine server](#) on page 57

Server Proxy User

Attribute

ServerProxyUser (spu)

Purpose

Specifies the user name needed to connect to the proxy server used by the SQL engine server.

Valid Values

string

where:

string

Is the default user ID that is used to connect to the proxy server used by the SQL engine server.

Default Value

No default value

See also

[Using the SQL engine server](#) on page 57

Service URL

Attribute

ServiceURL (surl)

Purpose

Specifies the endpoint for the Microsoft SharePoint service to which you are connecting.

Valid Values

String

where:

String

is the Microsoft SharePoint endpoint.

Default Value

No default value

Statement Call Limit

Attribute

StmtCallLimit (scl)

Purpose

Specifies the maximum number of web service calls the driver can make when executing any single SQL statement or metadata query.

Valid Values

0 | x

where:

x is a positive integer that defines the maximum number of web service calls up to 2147483647 the driver can make when executing any single SQL statement or metadata query.

Behavior

If set to 0, there is no limit.

If set to x , the driver uses this value to set the maximum number of web service calls on a single connection that can be made when executing a SQL statement. This limit can be overridden by changing the `STMT_CALL_LIMIT` session attribute using the `ALTER SESSION` statement. For example, the following statement sets the statement call limit to 10 web service calls:

```
ALTER SESSION SET STMT_CALL_LIMIT=10
```

If the web service call limit is exceeded, the behavior of the driver depends on the value specified for the Statement Call Limit Behavior (StmtCallLimitBehavior) option.

Default Value

0

See also

[Statement Call Limit Behavior](#) on page 92

Statement Call Limit Behavior

Attribute

StmtCallLimitBehavior (sclb)

Purpose

Specifies the behavior of the driver when the maximum web service call limit specified by the Statement Call Limit (StmtCallLimit) option is exceeded.

Valid Values

0 | 1

Behavior

If set to 0 (ReturnResults), the driver returns any partial results it received prior to the call limit being exceeded. The driver generates a warning that not all of the results were fetched.

If set to 1 (ErrorAlways), the driver generates an exception if the maximum web service call limit is exceeded.

Default Value

1 (ErrorAlways)

See also

[Statement Call Limit](#) on page 92

SQL Engine Mode

Attribute

SQLEngineMode (sem)

Purpose

Specifies whether the driver's SQL engine runs in the same process as the driver (direct mode) or runs in a process that is separate from the driver (server mode). You must be an administrator to modify the server mode configuration values, and to start or stop the SQL engine service.

Valid Values

0 | 1 | 2

Behavior

If set to 0 (Auto), the SQL engine attempts to run in server mode first; however, if server mode is unavailable, it runs in direct mode. To use server mode with this value, you must start the SQL Engine service before using the driver (see "Using the SQL engine server" for more information).

If set to 1 (Server), the SQL engine runs in server mode. The SQL engine operates in a separate process from the driver within its own JVM. You must start the SQL Engine service before using the driver (see "Using the SQL engine server" for more information).

If set to 2 (Direct), the SQL engine runs in direct mode. The driver and its SQL engine run in a single process within the same JVM.

Important: Changes you make to the server mode configuration affect all DSNs sharing the service.

Default Value

For Windows:

1 (Server)

For Linux:

2 (Direct)

See also

[Using the SQL engine server](#) on page 57

SQL Service

Attribute

SQLService (ss)

Purpose

Displays the name of the ODBC SQL engine service that runs as a separate process instead of being loaded within the process of an ODBC application.

Note: This option is used only for display purposes in the configuration manager. No value should be specified for this option.

Default Value

No default value

See also

[Using the SQL engine server](#) on page 57

Token URI

Attribute

TokenURI (o2tu)

Purpose

Specifies the endpoint for retrieving access tokens when OAuth 2.0 authentication is enabled.

Valid Values

String

where:

String

is the endpoint used to retrieve access tokens.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

No default value

Transaction Mode

Attribute

TransactionMode (tm)

Purpose

Specifies how the driver handles manual transactions.

Valid Values

0 | 1 | 3

Behavior

If set to 0 (NoTransactions), the data source and the driver do not support transactions. Metadata indicates that the driver does not support transactions.

If set to 1 (Ignore), the data source does not support transactions and the driver always operates in auto-commit mode. Calls to set the driver to manual commit mode and to commit transactions are ignored. Calls to rollback a transaction cause the driver to throw an exception indicating that no transaction is started. Metadata indicates that the driver supports transactions and the ReadUncommitted transaction isolation level.

If set to 3 (Full), the driver passes explicit transaction control commands such as `BEGIN TRANSACTION`, `COMMIT`, and `ROLLBACK` to the service.

Default Value

3 (Full)

Web Service Compress Data

Attribute

WSCompressData (wscd)

Purpose

Specifies whether the driver compresses data it sends to or receives from a Microsoft SharePoint service.

Valid Values

0 | 1

Behavior

If set to 0 (Compress), the driver sends and receives compressed data to and from the service.

If set to 1 (None), the driver sends and receives uncompressed data to and from the service.

Notes

Setting this option to 1 (None) can significantly degrade performance.

Default Value

0 (Compress)

6

Supported SQL statements and extensions

The driver provides support for the SQL statements and the SQL extensions described in this section. SQL extensions are denoted by an (EXT) in the topic title.

For details, see the following topics:

- [Alter Session \(EXT\)](#)
- [Delete](#)
- [Explain Plan](#)
- [Insert](#)
- [Refresh Map \(EXT\)](#)
- [Select](#)
- [Update](#)
- [Subqueries](#)
- [SQL expressions](#)

Alter Session (EXT)

Purpose

Changes various attributes of a local or remote session. A local session maintains the state of the overall connection. A remote session maintains the state that pertains to a particular remote data source connection.

Syntax

```
ALTER SESSION SET attribute_name=value
```

where:

attribute_name

specifies the name of the attribute to be changed. Attributes apply to either local or remote sessions.

value

specifies the value for that attribute.

The following table lists the local and remote session attributes, and provides descriptions of each.

Table 10: Alter Session Attributes

Attribute Name	Session Type	Description
Current_Schema	Local	Sets the current schema for the local session. The current schema is the schema used when an identifier in a SQL statement is unqualified. The string value must be the name of a schema visible in the local session. For example: <pre>ALTER SESSION SET CURRENT_SCHEMA=MICROSOFT SHAREPOINT</pre>

Attribute Name	Session Type	Description
Stmt_Call_Limit	Local	<p>Sets the maximum number of Web service calls the driver can make in executing a statement. Setting the Stmt_Call_Limit attribute has the same effect as setting the Statement Call Limit connection option. It sets the default Web service call limit used by any statement on the connection. Executing this command on a statement overrides the previously set Statement Call Limit for the connection. The value specified must be a positive integer or 0. The value 0 means that no call limit exists. For example:</p> <pre>ALTER SESSION SET STMT_CALL_LIMIT=150</pre>
Ws_Call_Count	Remote	<p>Resets the Web service call count of a remote session to the value specified. The value must be 0 or a positive integer. WS_Call_Count represents the total number of Web service calls made to the remote data source instance for the current session. For example:</p> <pre>ALTER SESSION SET microsoft sharepoint.WS_CALL_COUNT=0</pre> <p>The current value of WS_Call_Count can be obtained by referring to the System_Remote_Sessions system table (see SYSTEM_REMOTE_SESSIONS Catalog Table for details). For example:</p> <pre>SELECT * from information_schema.system_remote_sessions WHERE session_id = cursessionid()</pre>

Delete

Purpose

The Delete statement is used to delete rows from a table.

Syntax

```
DELETE FROM table_name [WHERE search_condition]
```

where:

table_name

specifies the name of the table from which you want to delete rows.

search_condition

is an expression that identifies which rows to delete from the table.

Notes

- The Where clause determines which rows are to be deleted. Without a Where clause, all rows of the table are deleted, but the table is left intact. See "Where Clause" for information about the syntax of Where clauses. Where clauses can contain subqueries.

Example A

This example shows a Delete statement on the emp table.

```
DELETE FROM emp WHERE emp_id = 'E10001'
```

Each Delete statement removes every record that meets the conditions in the Where clause. In this case, every record having the employee ID E10001 is deleted. Because employee IDs are unique in the employee table, at most, one record is deleted.

Example B

This example shows using a subquery in a Delete clause.

```
DELETE FROM emp WHERE dept_id = (SELECT dept_id FROM dept WHERE dept_name = 'Marketing')
```

The records of all employees who belong to the department named Marketing are deleted.

Notes

- To enable Insert, Update, and Delete, set the ReadOnly connection option to *false*.

Explain Plan

Purpose

Retrieves a detailed list of the elements in the execution plan. It generates a result set with a single column named OPERATION. The individual elements that comprise the plan are returned as rows in the result set.

Syntax

```
EXPLAIN PLAN FOR {SELECT ... | DELETE ... | INSERT ... | UPDATE ...}
```

The returned list of elements includes the indexes used for performing the query and can be used to optimize the query.

Insert

Purpose

The Insert statement is used to add new rows to a local table. You can specify either of the following options:

- List of values to be inserted as a new row
- Select statement that copies data from another table to be inserted as a set of new rows

Syntax

```
INSERT INTO table_name [(column_name[,column_name]...)] {VALUES (expression
[,expression]...) | select_statement}
```

table_name

is the name of the table in which you want to insert rows.

column_name

is optional and specifies an existing column. Multiple column names (a column list) must be separated by commas. A column list provides the name and order of the columns, the values of which are specified in the Values clause. If you omit a *column_name* or a column list, the value expressions must provide values for all columns defined in the table and must be in the same order that the columns are defined for the table. Table columns that do not appear in the column list are populated with the default value, or with NULL if no default value is specified.

expression

is the list of expressions that provides the values for the columns of the new record. Typically, the expressions are constant values for the columns. Character string values must be enclosed in single quotation marks ('). See "Literals" for more information.

select_statement

is a query that returns values for each *column_name* value specified in the column list. Using a Select statement instead of a list of value expressions lets you select a set of rows from one table and insert it into another table using a single Insert statement. The Select statement is evaluated before any values are inserted. This query cannot be made on the table into which values are inserted. See "Select" for information about Select statements.

Notes

- To enable Insert, Update, and Delete, set the ReadOnly connection option to *false*.

Specifying an external ID column

Use the following syntax to specify an external ID column to look up the value of a foreign key column.

Syntax

```
column_name EXT_ID [schema_name.[table_name.] ]ext_id_column
```

where:

EXT_ID

is used to specify that the column specified by *ext_id_column* is used to look up the rowid to be inserted into the column specified by *column_name*.

schema_name

is the name of the schema of the table that contains the foreign key column being specified as the external ID column.

table_name

is the name of the table that contains the foreign key column being specified as the external ID column.

ext_id_column

is the external ID column.

Example A

This example uses a list of expressions to insert records. Each Insert statement adds one record to the table. In this case, one record is added to the table `emp`. Values are specified for five columns. The remaining columns in the table are assigned the default value or NULL if no default value is specified.

```
INSERT INTO emp (last_name,
                 first_name,
                 emp_id,
                 salary,
                 hire_date)
VALUES ('Smith', 'John', 'E22345', 27500, {1999-04-06})
```

Example B

This example uses a Select statement to insert records. The number of columns in the result of the Select statement must match exactly the number of columns in the table if no column list is specified, or it must match the number of column names specified in the column list. A new entry is created in the table for every row of the Select result.

```
INSERT INTO emp1 (first_name,
                 last_name,
                 emp_id,
                 dept,
                 salary)
SELECT first_name, last_name, emp_id, dept, salary FROM emp
WHERE dept = 'D050'
```

Example C

This example uses a list of expressions to insert records and specifies an external ID column (a foreign key column) named `accountId` that references a table that has an external ID column named `AccountNum`.

```
INSERT INTO emp (last_name,
                 first_name,
                 emp_id,
                 salary,
                 hire_date,
                 accountId EXT_ID AccountNum)
VALUES ('Smith', 'John', 'E22345', 27500, {1999-04-06}, 0001)
```

Refresh Map (EXT)

Purpose

The REFRESH MAP statement adds newly discovered objects to your relational view of native data. It also incorporates any configuration changes made to your relational view by reloading the schema definition and associated files.

Syntax

REFRESH MAP

Notes

- REFRESH MAP is an expensive query since it involves the discovery of native data.

Select

Purpose

Use the Select statement to fetch results from one or more tables.

Syntax

```
SELECT select_clause from_clause
[where_clause]
[groupby_clause]
[having_clause]
[ {UNION [ALL | DISTINCT] |
  {MINUS [DISTINCT] | EXCEPT [DISTINCT]} |
  INTERSECT [DISTINCT]} select_statement ]
[limit_clause]
```

where:

select_clause

specifies the columns from which results are to be returned by the query. See "Select clause" for a complete explanation.

from_clause

specifies one or more tables on which the other clauses in the query operate. See "From clause" for a complete explanation.

where_clause

is optional and restricts the results that are returned by the query. See "Where clause" for a complete explanation.

groupby_clause

is optional and allows query results to be aggregated in terms of groups. See "Group By clause" for a complete explanation.

having_clause

is optional and specifies conditions for groups of rows (for example, display only the departments that have salaries totaling more than \$200,000). See "Having clause" for a complete explanation.

UNION

is an optional operator that combines the results of the left and right Select statements into a single result. See "Union operator" for a complete explanation.

INTERSECT

is an optional operator that returns a single result by keeping any distinct values from the results of the left and right Select statements. See "Intersect operator" for a complete explanation.

EXCEPT | MINUS

are synonymous optional operators that returns a single result by taking the results of the left Select statement and removing the results of the right Select statement. See "Except and Minus operators" for a complete explanation.

orderby_clause

is optional and sorts the results that are returned by the query. See "Order By clause" for a complete explanation.

limit_clause

is optional and places an upper bound on the number of rows returned in the result. See "Limit clause" for a complete explanation.

Select clause

Purpose

Use the Select clause to specify with a list of column expressions that identify columns of values that you want to retrieve or an asterisk (*) to retrieve the value of all columns.

Syntax

```
SELECT [{LIMIT offsetnumber | TOP number}] [ALL | DISTINCT] {* | column_expression
[[AS] column_alias] [,column_expression [[AS] column_alias], ...}
```

where:

`LIMIT offset number`

creates the result set for the Select statement first and then discards the first number of rows specified by *offset* and returns the number of remaining rows specified by *number*. To not discard any of the rows, specify 0 for *offset*, for example, `LIMIT 0 number`. To discard the first *offset* number of rows and return all the remaining rows, specify 0 for *number*, for example, `LIMIT offset 0`.

`TOP number`

is equivalent to `LIMIT 0 number`.

column_expression

can be simply a column name (for example, `last_name`). More complex expressions may include mathematical operations or string manipulation (for example, `salary * 1.05`). See "SQL expressions" for details. *column_expression* can also include aggregate functions. See "Aggregate functions" for details.

column_alias

can be used to give the column a descriptive name. For example, to assign the alias department to the column dep:

```
SELECT dep AS department FROM emp
```

DISTINCT

eliminates duplicate rows from the result of a query. This operator can precede the first column expression. For example:

```
SELECT DISTINCT dep FROM emp
```

Notes

- Separate multiple column expressions with commas (for example, `SELECT last_name, first_name, hire_date`).
- Column names can be prefixed with the table name or table alias. For example, `SELECT emp.last_name` or `e.last_name`, where `e` is the alias for the table `emp`.
- NULL values are not treated as distinct from each other. The default behavior is that all result rows be returned, which can be made explicit with the keyword `ALL`.

See also

[SQL expressions](#) on page 116

Aggregate functions

Aggregate functions can also be a part of a Select clause. Aggregate functions return a single value from a set of rows. An aggregate can be used with a column name (for example, `AVG(salary)`) or in combination with a more complex column expression (for example, `AVG(salary * 1.07)`).

The following table lists supported aggregate functions.

Note: Doubly nested aggregates, such as `SUM(COUNT(col1))`, are currently not permitted by the driver.

Table 11: Aggregate Functions

Aggregate	Returns
AVG	The average of the values in a numeric column expression. For example, <code>AVG(salary)</code> returns the average of all salary column values.
COUNT	<p>The number of values in any field expression. For example, <code>COUNT(name)</code> returns the number of name values. When using <code>COUNT</code> with a field name, <code>COUNT</code> returns the number of non-NULL column values. A special example is <code>COUNT(*)</code>, which returns the number of rows in the set, including rows with NULL values.</p> <hr/> <p>Note: The driver does not support <code>COUNT(DISTINCT *)</code>. For example, <code>SELECT COUNT(DISTINCT *) FROM mytable</code> results in a syntax error.</p> <hr/>

MAX	The maximum value in any column expression. For example, MAX(salary) returns the maximum salary column value.
MIN	The minimum value in any column expression. For example, MIN(salary) returns the minimum salary column value.
SUM	The total of the values in a numeric column expression. For example, SUM(salary) returns the sum of all salary column values.

Example

The following example uses the COUNT, MAX, and AVG aggregate functions:

```
SELECT
    COUNT(amount) AS numOpportunities,
    MAX(amount) AS maxAmount,
    AVG(amount) AS avgAmount
FROM opportunity o INNER JOIN user u
    ON o.ownerId = u.id
WHERE o.isClosed = 'false' AND
    u.name = 'MyName'
```

From clause

Purpose

The From clause indicates the tables to be used in the Select statement.

Syntax

```
FROM table_name [table_alias] [, ...]
```

where:

table_name

is the name of a table or a subquery. Multiple tables define an implicit inner join among those tables. Multiple table names must be separated by a comma. For example:

```
SELECT * FROM emp, dep
```

Subqueries can be used instead of table names. Subqueries must be enclosed in parentheses. See "Subquery in a From clause" for an example.

table_alias

is a name used to refer to a table in the rest of the Select statement. When you specify an alias for a table, you can prefix all column names of that table with the table alias.

Example

The following example specifies two table aliases, e for emp and d for dep:

```
SELECT e.name, d.deptName
FROM emp e, dep d
WHERE e.deptId = d.id
```

table_alias is a name used to refer to a table in the rest of the Select statement. When you specify an alias for a table, you can prefix all column names of that table with the table alias. For example, given the table specification:

```
FROM emp E
```

you may refer to the `last_name` field as `E.last_name`. Table aliases must be used if the Select statement joins a table to itself. For example:

```
SELECT * FROM emp E, emp F WHERE E.mgr_id = F.emp_id
```

The equal sign (=) includes only matching rows in the results.

Join in a From clause

Purpose

You can use a Join as a way to associate multiple tables within a Select statement. Joins may be either explicit or implicit. For example, the following is the example from the previous section restated as an explicit inner join:

```
SELECT * FROM emp INNER JOIN dep ON id=empId
SELECT e.name, d.deptName
FROM emp e INNER JOIN dep d ON e.deptId = d.id;
```

whereas the following is the same statement as an implicit inner join:

```
SELECT * FROM emp, dep WHERE emp.deptID=dep.id
```

Note: The ON clause in a join expression must evaluate to a true or false value.

Syntax

```
FROM table_name {RIGHT OUTER | INNER | LEFT OUTER | CROSS | FULL OUTER} JOIN table.key
ON search-condition
```

Example

In this example, two tables are joined using `LEFT OUTER JOIN`. T1, the first table named includes nonmatching rows.

```
SELECT * FROM T1 LEFT OUTER JOIN T2 ON T1.key = T2.key
```

If you use a `CROSS JOIN`, no ON expression is allowed for the join.

Subquery in a From clause

Subqueries can be used in the From clause in place of table references (*table_name*).

Example

```
SELECT * FROM (SELECT * FROM emp WHERE sal > 10000) new_emp, dept WHERE
new_emp.deptno = dept.deptno
```

See also

[Subqueries](#) on page 114

Where clause

Purpose

Specifies the conditions that rows must meet to be retrieved.

Syntax

```
WHERE expr1 rel_operator expr2
```

where:

expr1

is either a column name, literal, or expression.

expr2

is either a column name, literal, expression, or subquery. Subqueries must be enclosed in parentheses.

rel_operator

is the relational operator that links the two expressions.

Example

The following Select statement retrieves the first and last names of employees that make at least \$20,000.

```
SELECT last_name, first_name FROM emp WHERE salary >= 20000
```

See also

[SQL expressions](#) on page 116

[Subqueries](#) on page 114

Group By clause

Purpose

Specifies the names of one or more columns by which the returned values are grouped. This clause is used to return a set of aggregate values.

Syntax

```
GROUP BY column_expression [,...]
```

where:

column_expression

is either a column name or a SQL expression. Multiple values must be separated by a comma. If *column_expression* is a column name, it must match one of the column names specified in the Select clause. Also, the Group By clause must include all non-aggregate columns specified in the Select list.

Example

The following example totals the salaries in each department:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

See also

[SQL expressions](#) on page 116

[Subqueries](#) on page 114

Having clause

Purpose

Specifies conditions for groups of rows (for example, display only the departments that have salaries totaling more than \$200,000). This clause is valid only if you have already defined a Group By clause.

Syntax

```
HAVING expr1 rel_operator expr2
```

where:

```
expr1 | expr2
```

can be column names, constant values, or expressions. These expressions do not have to match a column expression in the Select clause. See "SQL expressions" for details regarding SQL expressions.

```
rel_operator
```

is the relational operator that links the two expressions.

Example

The following example returns only the departments that have salaries totaling more than \$200,000:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id HAVING sum(salary) > 200000
```

See also

[SQL expressions](#) on page 116

[Subqueries](#) on page 114

Union operator

Purpose

Combines the results of two Select statements into a single result. The single result is all the returned rows from both Select statements. By default, duplicate rows are not returned. To return duplicate rows, use the All keyword (`UNION ALL`).

Syntax

```
select_statement  
UNION [ALL | DISTINCT] | {MINUS [DISTINCT] | EXCEPT [DISTINCT]} | INTERSECT  
[DISTINCT]select_statement
```

Notes

- When using the Union operator, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp  
UNION  
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (*salary* FROM *emp* has a different data type than *last_name* FROM *raises*). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp  
UNION  
SELECT salary, last_name FROM raises
```

Intersect operator

Purpose

Intersect operator returns a single result set. The result set contains rows that are returned by both Select statements. Duplicates are returned unless the Distinct operator is added.

Syntax

```
select_statement  
INTERSECT [DISTINCT]  
select_statement
```

where:

DISTINCT

eliminates duplicate rows from the results.

Notes

- When using the Intersect operator, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
INTERSECT [DISTINCT]
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
INTERSECT
SELECT salary, last_name FROM raises
```

Except and Minus operators

Purpose

Return the rows from the left Select statement that are not included in the result of the right Select statement.

Syntax

```
select_statement
{EXCEPT [DISTINCT] | MINUS [DISTINCT]}
select_statement
```

where:

DISTINCT

eliminates duplicate rows from the results.

Notes

- When using one of these operators, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
EXCEPT
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
EXCEPT
SELECT salary, last_name FROM raises
```

Order By clause

Purpose

The Order By clause specifies how the rows are to be sorted.

Syntax

```
ORDER BY sort_expression [DESC | ASC] [,...]
```

where:

sort_expression

is either the name of a column, a column alias, a SQL expression, or the positioned number of the column or expression in the select list to use.

The default is to perform an ascending (ASC) sort.

Example

To sort by `last_name` and then by `first_name`, you could use either of the following Select statements:

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY last_name, first_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY 2,3
```

In the second example, `last_name` is the second item in the Select list, so `ORDER BY 2,3` sorts by `last_name` and then by `first_name`.

See also

[SQL expressions](#) on page 116

Limit clause

Purpose

Places an upper bound on the number of rows returned in the result.

Syntax

```
LIMIT number_of_rows [OFFSET offset_number]
```

where:

number_of_rows

specifies a maximum number of rows in the result. A negative number indicates no upper bound.

OFFSET

specifies how many rows to skip at the beginning of the result set. *offset_number* is the number of rows to skip.

Notes

- In a compound query, the Limit clause can appear only on the final Select statement. The limit is applied to the entire query, not to the individual Select statement to which it is attached.

Example

The following example returns a maximum of 20 rows.

```
SELECT last_name, first_name FROM emp WHERE salary > 20000 ORDER BY dept_idc LIMIT
20
```

Update

Purpose

An Update statement changes the value of columns in the selected rows of a table.

Syntax

```
UPDATE table_name SET column_name = expression
[, column_name = expression] [WHERE conditions]
```

table_name

is the name of the table for which you want to update values.

column_name

is the name of a column, the value of which is to be changed. Multiple column values can be changed in a single statement.

expression

is the new value for the column. The expression can be a constant value or a subquery that returns a single value. Subqueries must be enclosed in parentheses.

Example A

The following example changes every record that meets the conditions in the Where clause. In this case, the salary and exempt status are changed for all employees having the employee ID E10001. Because employee IDs are unique in the emp table, only one record is updated.

```
UPDATE emp SET salary=32000, exempt=1
```

```
WHERE emp_id = 'E10001'
```

Example B

The following example uses a subquery. In this example, the salary is changed to the average salary in the company for the employee having employee ID E10001.

```
UPDATE emp SET salary = (SELECT avg(salary) FROM emp)
WHERE emp_id = 'E10001'
```

Notes

- To enable Insert, Update, and Delete, set the ReadOnly connection option to `false`.
- A Where clause can be used to restrict which rows are updated.

Subqueries

A query is an operation that retrieves data from one or more tables or views. In this reference, a top-level query is called a Select statement, and a query nested within a Select statement is called a subquery.

A subquery is a query expression that appears in the body of another expression such as a Select, an Update, or a Delete statement. In the following example, the second Select statement is a subquery:

```
SELECT * FROM emp WHERE deptno IN (SELECT deptno FROM dept)
```

IN predicate

Purpose

The In predicate specifies a set of values against which to compare a result set. If the values are being compared against a subquery, only a single column result set is returned.

Syntax

```
value [NOT] IN (value1, value2,...)
```

OR

```
value [NOT] IN (subquery)
```

Example

```
SELECT * FROM emp WHERE deptno IN
(SELECT deptno FROM dept WHERE dname <> 'Sales')
```

EXISTS predicate

Purpose

The Exists predicate is true only if the cardinality of the subquery is greater than 0; otherwise, it is false.

Syntax

```
EXISTS (subquery)
```

Example

```
SELECT empno, ename, deptno FROM emp e WHERE EXISTS
(SELECT deptno FROM dept WHERE e.deptno = dept.deptno)
```

UNIQUE predicate

Purpose

The Unique predicate is used to determine whether duplicate rows exist in a virtual table (one returned from a subquery).

Syntax

```
UNIQUE (subquery)
```

Example

```
SELECT * FROM dept d WHERE UNIQUE
(SELECT deptno FROM emp e WHERE e.deptno = d.deptno)
```

Correlated subqueries

Purpose

A correlated subquery is a subquery that references a column from a table referred to in the parent statement. A correlated subquery is evaluated once for each row processed by the parent statement. The parent statement can be a Select, Update, or Delete statement.

A correlated subquery answers a multiple-part question in which the answer depends on the value in each row processed by the parent statement. For example, you can use a correlated subquery to determine which employees earn more than the average salaries for their departments. In this case, the correlated subquery specifically computes the average salary for each department.

Syntax

```
SELECT select_list
FROM table1 t_alias1
WHERE expr rel_operator
      (SELECT column_list
       FROM table2 t_alias2
       WHERE t_alias1.columnrel_operatort_alias2.column)
UPDATE table1 t_alias1
SET column =
      (SELECT expr
       FROM table2 t_alias2
       WHERE t_alias1.column = t_alias2.column)
DELETE FROM table1 t_alias1
WHERE column rel_operator
      (SELECT expr
```

```
FROM table2 t_alias2
WHERE t_alias1.column = t_alias2.column)
```

Notes

- Correlated column names in correlated subqueries must be explicitly qualified with the table name of the parent.

Example A

The following statement returns data about employees whose salaries exceed their department average. This statement assigns an alias to `emp`, the table containing the salary information, and then uses the alias in a correlated subquery:

```
SELECT deptno, ename, sal FROM emp x WHERE sal >
(SELECT AVG(sal) FROM emp WHERE x.deptno = deptno)
ORDER BY deptno
```

Example B

This is an example of a correlated subquery that returns row values:

```
SELECT * FROM dept "outer" WHERE 'manager' IN
(SELECT managername FROM emp
WHERE "outer".deptno = emp.deptno)
```

Example C

This is an example of finding the department number (`deptno`) with multiple employees:

```
SELECT * FROM dept main WHERE 1 <
(SELECT COUNT(*) FROM emp WHERE deptno = main.deptno)
```

Example D

This is an example of correlating a table with itself:

```
SELECT deptno, ename, sal FROM emp x WHERE sal >
(SELECT AVG(sal) FROM emp WHERE x.deptno = deptno)
```

SQL expressions

An expression is a combination of one or more values, operators, and SQL functions that evaluate to a value. You can use expressions in the `Where`, and `Having` of `Select` statements; and in the `Set` clauses of `Update` statements.

Expressions enable you to use mathematical operations as well as character string manipulation operators to form complex queries.

The driver supports both unquoted and quoted identifiers. An unquoted identifier must start with an ASCII alpha character and can be followed by zero

Quoted identifiers must be enclosed in double quotation marks ("""). A quoted identifier can contain any Unicode character including the space character. The driver recognizes the Unicode escape sequence `\uxxxx` as a Unicode character. You can specify a double quotation mark in a quoted identifier by escaping it with a double quotation mark.

The maximum length of both quoted and unquoted identifiers is 128 characters.

Valid expression elements are:

- Column names
- Literals
- Operators
- Functions

Column names

The most common expression is a simple column name. You can combine a column name with other expression elements.

Literals

Literals are fixed data values. For example, in the expression `PRICE * 1.05`, the value 1.05 is a constant. Literals are classified into types, including the following:

- Binary
- Character string
- Date
- Floating point
- Integer
- Numeric
- Time
- Timestamp

The following table describes the literal format for supported SQL data types.

Table 12: Literal Syntax Examples

SQL Type	Literal Syntax	Example
BIGINT	<i>n</i> where <i>n</i> is any valid integer value in the range of the INTEGER data type	12 or -34 or 0
BOOLEAN	Min Value: 0 Max Value: 1	0 1
DATE	DATE' <i>date</i> '	'2010-05-21'
DATETIME	TIMESTAMP' <i>ts</i> '	'2010-05-21 18:33:05.025'

SQL Type	Literal Syntax	Example
DECIMAL	$n.f$ where: n is the integral part f is the fractional part	0.25 3.1415 -7.48
DOUBLE	$n.fEx$ where: n is the integral part f is the fractional part x is the exponent	1.2E0 or 2.5E40 or -3.45E2 or 5.67E-4
INTEGER	n where n is a valid integer value in the range of the INTEGER data type	12 or -34 or 0
LONGVARBINARY	' <i>hex_value</i> '	'000482ff'
LONGVARCHAR	' <i>value</i> '	'This is a string literal'
TIME	TIME' <i>time</i> '	'2010-05-21 18:33:05.025'
VARCHAR	' <i>value</i> '	'This is a string literal'

Character string literals

Text specifies a character string literal. A character string literal must be enclosed in single quotation marks. To represent one single quotation mark within a literal, you must enter two single quotation marks. When the data in the fields is returned to the client, trailing blanks are stripped.

A character string literal can have a maximum length of 32 KB, that is, (32*1024) bytes.

Example

```
'Hello'  
'Jim''s friend is Joe'
```

Numeric literals

Unquoted numeric values are treated as numeric literals. If the unquoted numeric value contains a decimal point or exponent, it is treated as a real literal; otherwise, it is treated as an integer literal.

Example`+1894.1204`**Binary literals**

Binary literals are represented with single quotation marks. The valid characters in a binary literal are 0-9, a-f, and A-F.

Example`'00af123d'`**Date/Time literals**

Date and time literal values are enclosed in single quotation marks (*'value'*).

- The format for a Date literal is DATE'*date*'.
- The format for a Time literal is TIME'*time*'.
- The format for a Timestamp literal is TIMESTAMP'*ts*'.

Integer literals

Integer literals are represented by a string of numbers that are not enclosed in quotation marks and do not contain decimal points.

Notes

- Integer constants must be whole numbers; they cannot contain decimals.
- Integer literals can start with sign characters (+/-).

Example`1994 or -2`**Operators**

This section describes the operators that can be used in SQL expressions.

Note: Numeric operators are restricted to numeric types. Numeric operators do not support non-numeric types.

Unary operator

A unary operator operates on only one operand.

operator operand

Binary operator

A binary operator operates on two operands.

operand1 operator operand2

If an operator is given a null operand, the result is always null. The only operator that does not follow this rule is concatenation (||).

Arithmetic operators

You can use an arithmetic operator in an expression to negate, add, subtract, multiply, and divide numeric values. The result of this operation is also a numeric value. The + and - operators are also supported in date/time fields to allow date arithmetic. The following table lists the supported arithmetic operators.

Table 13: Arithmetic Operators

Operator	Purpose	Example
+ -	Denotes a positive or negative expression. These are unary operators.	SELECT * FROM emp WHERE comm = -1
* /	Multiplies, divides. These are binary operators.	UPDATE emp SET sal = sal + sal * 0.10
+ -	Adds, subtracts. These are binary operators.	SELECT sal + comm FROM emp WHERE empno > 100

Concatenation operator

The concatenation operator manipulates character strings. The following table lists the only supported concatenation operator.

Table 14: Concatenation Operator

Operator	Purpose	Example
	Concatenates character strings.	SELECT 'Name is' ename FROM emp

The result of concatenating two character strings is the data type VARCHAR.

Comparison operators

Comparison operators compare one expression to another. The result of such a comparison can be TRUE, FALSE, or UNKNOWN (if one of the operands is NULL). The driver considers the UNKNOWN result as FALSE.

The following table lists the supported comparison operators.

Table 15: Comparison Operators

Operator	Purpose	Example
=	Equality test.	SELECT * FROM emp WHERE sal = 1500
!=<>	Inequality test.	SELECT * FROM emp WHERE sal != 1500

Operator	Purpose	Example
><	"Greater than" and "less than" tests.	SELECT * FROM emp WHERE sal > 1500 SELECT * FROM emp WHERE sal < 1500
>=<=	"Greater than or equal to" and "less than or equal to" tests.	SELECT * FROM emp WHERE sal >= 1500 SELECT * FROM emp WHERE sal <= 1500
LIKE	% and _ wildcards can be used to search for a pattern in a column. The percent sign denotes zero, one, or multiple characters, while the underscore denotes a single character. The right-hand side of a LIKE expression must evaluate to a string or binary.	SELECT * FROM emp WHERE ENAME LIKE 'J%'
ESCAPE clause in LIKE operator LIKE 'pattern string' ESCAPE 'c'	The Escape clause is supported in the LIKE predicate to indicate the escape character. Escape characters are used in the pattern string to indicate that any wildcard character that is after the escape character in the pattern string should be treated as a regular character. The default escape character is backslash (\).	SELECT * FROM emp WHERE ENAME LIKE 'J%_%' ESCAPE '\' This matches all records with names that start with letter 'J' and have the '_' character in them. SELECT * FROM emp WHERE ENAME LIKE 'JOE_JOHN' ESCAPE '\' This matches only records with name 'JOE_JOHN'.
[NOT] IN	"Equal to any member of" test.	SELECT * FROM emp WHERE job IN ('CLERK', 'ANALYST') SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30)
[NOT] BETWEEN x AND y	"Greater than or equal to x" and "less than or equal to y."	SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000
EXISTS	Tests for existence of rows in a subquery.	SELECT empno, ename, deptno FROM emp e WHERE EXISTS (SELECT deptno FROM dept WHERE e.deptno = dept.deptno)
IS [NOT] NULL	Tests whether the value of the column or expression is NULL.	SELECT * FROM emp WHERE ename IS NOT NULL SELECT * FROM emp WHERE ename IS NULL

Logical operators

A logical operator combines the results of two component conditions to produce a single result or to invert the result of a single condition. The following table lists the supported logical operators.

Table 16: Logical Operators

Operator	Purpose	Example
NOT	Returns TRUE if the following condition is FALSE. Returns FALSE if it is TRUE. If it is UNKNOWN, it remains UNKNOWN.	<pre>SELECT * FROM emp WHERE NOT (job IS NULL) SELECT * FROM emp WHERE NOT (sal BETWEEN 1000 AND 2000)</pre>
AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise, returns UNKNOWN.	<pre>SELECT * FROM emp WHERE job = 'CLERK' AND deptno = 10</pre>
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE; otherwise, returns UNKNOWN.	<pre>SELECT * FROM emp WHERE job = 'CLERK' OR deptno = 10</pre>

Example

In the Where clause of the following Select statement, the AND logical operator is used to ensure that managers earning more than \$1000 a month are returned in the result:

```
SELECT * FROM emp WHERE jobtitle = manager AND sal > 1000
```

Operator precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. The following table shows the order in which the operators are evaluated. The operators in the first line are evaluated first, then those in the second line, and so on. Operators in the same line are evaluated left to right in the expression. You can change the order of precedence by using parentheses. Enclosing expressions in parentheses forces them to be evaluated together.

Table 17: Operator Precedence

Precedence	Operator
1	+ (Positive), - (Negative)
2	*(Multiply), / (Division)
3	+ (Add), - (Subtract)
4	(Concatenate)
5	=, >, <, >=, <=, <>, != (Comparison operators)
6	NOT, IN, LIKE

Precedence	Operator
7	AND
8	OR

Example A

The query in the following example returns employee records for which the department number is 1 or 2 and the salary is greater than \$1000:

```
SELECT * FROM emp WHERE (deptno = 1 OR deptno = 2) AND sal > 1000
```

Because parenthetical expressions are forced to be evaluated first, the OR operation takes precedence over AND.

Example B

In the following example, the query returns records for all the employees in department 1, but only employees whose salary is greater than \$1000 in department 2.

```
SELECT * FROM emp WHERE deptno = 1 OR deptno = 2 AND sal > 1000
```

The AND operator takes precedence over OR, so that the search condition in the example is equivalent to the expression `deptno = 1 OR (deptno = 2 AND sal > 1000)`.

Functions

The driver supports a number of functions that you can use in expressions, as listed and described in "Scalar functions."

Refer to "Scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for more information.

Conditions

A condition specifies a combination of one or more expressions and logical operators that evaluates to either TRUE, FALSE, or UNKNOWN. You can use a condition in the Where clause of the Delete, Select, and Update statements; and in the Having clauses of Select statements. The following describes supported conditions.

Table 18: Conditions

Condition	Description
Simple comparison	Specifies a comparison with expressions or subquery results. = , !=, <>, < , >, <=, >=
Group comparison	Specifies a comparison with any or all members in a list or subquery. [= , !=, <>, < , >, <=, >=] [ANY, ALL, SOME]

Condition	Description
Membership	Tests for membership in a list or subquery. [NOT] IN
Range	Tests for inclusion in a range. [NOT] BETWEEN
NULL	Tests for nulls. IS NULL, IS NOT NULL
EXISTS	Tests for existence of rows in a subquery. [NOT] EXISTS
LIKE	Specifies a test involving pattern matching. [NOT] LIKE
Compound	Specifies a combination of other conditions. CONDITION [AND/OR] CONDITION

Introduction to the SharePoint List data model

The driver exposes SharePoint lists as relational tables. SharePoint lists are created by users, and therefore the data model of a list depends on the columns added to the list by the user. The following topics include sample tables which show the columns that may appear in SharePoint lists and the associated data type for each.

- [DEFAULTCOLUMNSLIST](#) is a table that includes all the columns a user can specify in a list. The driver exposes all user-specified columns by default. The column names indicate the type of column created.
- [ALLCOLUMNSLIST](#) is a table that includes internal SharePoint columns, as well as user-specified columns. The internal columns are exposed when the AuditColumns connection property is set to `All`.

The following sections provide information on supported column types and how the driver exposes columns.

- [Modern column types](#)
- [Location complex types](#)
- [Hyperlink or Picture complex types](#)
- [Unfamiliar column names](#)

Modern column types

The driver supports SharePoint Modern column types as described in the following table.

Column type	Data type	Behavior
Single line of text	STRING(255)	Read-write

Column type	Data type	Behavior
Multiple lines of text	STRING(63999)	Read-write
Location	COMPLEX	Read-write
Number	DOUBLE	Read-write
Yes/No	BOOLEAN	Read-write
Person or Group	STRING(255)	Read-only
Date and Time	DATETIMEOFFSET(26)	Read-write
Choice	STRING(255)	Read-write
Hyperlink or Picture	COMPLEX	Read-only
Currency	DECIMAL(19)	Read-write

Location complex types

Location column types are complex columns that contain multiple fields. As shown here, the fields in Location columns are exposed by the driver as separate columns where *colname* is the name assigned to the column.

Column name	Data type	Behavior
<i>colname</i> _ADDRESS_COUNTRYORREGION	STRING(255)	Read-write
<i>colname</i> _ADDRESS_STATE	STRING(255)	Read-write
<i>colname</i> _ADDRESS_CITY	STRING(255)	Read-write
<i>colname</i> _ADDRESS_POSTALCODE	STRING(255)	Read-write
<i>colname</i> _ADDRESS_STREET	STRING(255)	Read-write
<i>colname</i> _COORDINATES_LATITUDE	DOUBLE	Read-write
<i>colname</i> _COORDINATES_LONGITUDE	DOUBLE	Read-write
<i>colname</i> _DISPLAYNAME	STRING(255)	Read-write

Hyperlink or Picture complex types

Hyperlink or Picture column types are complex columns that contain multiple fields. As shown in the following table, the two fields in Hyperlink or Picture columns are exposed by the driver as separate columns where *colname* is the name assigned to the column.

Column name	Data type	Behavior
<i>colname</i> _DESCRIPTION	STRING(255)	Read-write
<i>colname</i> _URL	STRING(255)	Read-write

Unfamiliar column names

When a naming conflict occurs, SharePoint generates a unique column name to replace the name specified when the column was created. In these cases, you may see an unfamiliar column name returned when performing a select operation from a table. For example, if the column name `PERCOL` conflicts with another column, SharePoint might create the unique column name `_X0076_H65`.

You can determine the original column name by searching for the unfamiliar column name in the native file. The native file is a metadata file that will include the original name and unfamiliar name. The location of the native file (as well as other internal files) is determined by the path specified for the SchemaMap connection property. The default location of the file depends on the platform you are using.

- For Windows platforms
 - User data source:
`user_profile\AppData\Local\Progress\DataDirect\SharePoint_Schema\user_name.native`
 - System data source:
`C:\Users\Default\AppData\Local\Progress\DataDirect\SharePoint_Schema\user_name.native`
- For UNIX/Linux
 - `~/progress/datadirect/sharepoint_schema/user_name.native`

The label field in the column definition will contain the original column name. For example:

```
<Field name="_X0076_H65" nativeType="STRING" filterable="0" label="PERCOL"
po4:navigation="fields"/>
```

For details, see the following topics:

- [ALLCOLUMNSLIST](#)
- [DEFAULTCOLUMNSLIST](#)

ALLCOLUMNSLIST

Columns

The ALLCOLUMNSLIST table includes internal SharePoint columns, as well as user-specified columns. The internal columns are exposed when the AuditColumns connection property is set to `All`. The ID column marked with an asterisk is the primary key.

Column Name	Data Type
ID*	INT32

Column Name	Data Type
_COMPLIANCEFLAGS	STRING(255)
_COMPLIANCETAG	STRING(255)
_COMPLIANCETAGUSERID	STRING(255)
_COMPLIANCETAGWRITTENTIME	STRING(255)
_ISRECORD	STRING(255)
_UIVERSIONSTRING	STRING(255)
APPAUTHOR	STRING(255)
APPEDITOR	STRING(255)
ATTACHMENTS	STRING(255)
AUTHOR	STRING(255)
BOOLEANCOL	BOOLEAN
CHOICECOL	STRING(255)
COMPLIANCEASSETID	STRING(255)
CONTENTTYPE	STRING(255)
CONTENTTYPE_ID	STRING(2000)
CONTENTTYPE_NAME	STRING(2000)
CREATED	DATETIMEOFFSET(26)
CREATEDBY_APPLICATION_DISPLAYNAME	STRING(2000)
CREATEDBY_APPLICATION_EMAIL	STRING(2000)
CREATEDBY_APPLICATION_ID	STRING(2000)
CREATEDBY_APPLICATION_USERPRINCIPALNAME	STRING(2000)
CREATEDBY_GROUP_DISPLAYNAME	STRING(2000)
CREATEDBY_GROUP_EMAIL	STRING(2000)
CREATEDBY_GROUP_ID	STRING(2000)
CREATEDBY_GROUP_USERPRINCIPALNAME	STRING(2000)
CREATEDBY_USER_DISPLAYNAME	STRING(2000)

Column Name	Data Type
CREATEDBY_USER_EMAIL	STRING(2000)
CREATEDBY_USER_ID	STRING(2000)
CREATEDBY_USER_USERPRINCIPALNAME	STRING(2000)
CREATEDDATETIME	DATETIMEOFFSET(26)
CURRENCYCOL	DECIMAL(19, 4)
DESCRIPTION	STRING(2000)
DOCICON	STRING(255)
EDIT	STRING(255)
EDITOR	STRING(255)
ETAG	STRING(2000)
FOLDERCHILDCOUNT	STRING(255)
ITEMCHILDCOUNT	STRING(255)
LASTMODIFIEDBY_APPLICATION_DISPLAYNAME	STRING(2000)
LASTMODIFIEDBY_APPLICATION_EMAIL	STRING(2000)
LASTMODIFIEDBY_APPLICATION_ID	STRING(2000)
LASTMODIFIEDBY_APPLICATION_USERPRINCIPALNAME	STRING(2000)
LASTMODIFIEDBY_GROUP_DISPLAYNAME	STRING(2000)
LASTMODIFIEDBY_GROUP_EMAIL	STRING(2000)
LASTMODIFIEDBY_GROUP_ID	STRING(2000)
LASTMODIFIEDBY_GROUP_USERPRINCIPALNAME	STRING(2000)
LASTMODIFIEDBY_USER_DISPLAYNAME	STRING(2000)
LASTMODIFIEDBY_USER_EMAIL	STRING(2000)
LASTMODIFIEDBY_USER_ID	STRING(2000)
LASTMODIFIEDBY_USER_USERPRINCIPALNAME	STRING(2000)
LASTMODIFIEDDATETIME	DATETIMEOFFSET(26)
LINKCOL_DESCRIPTION	STRING(255)

Column Name	Data Type
LINKCOL_URL	STRING(255)
LINKTITLE	STRING(255)
LINKTITLENOMENU	STRING(255)
LOCATIONCOL_ADDRESS_CITY	STRING(255)
LOCATIONCOL_ADDRESS_COUNTRYORREGION	STRING(255)
LOCATIONCOL_ADDRESS_POSTALCODE	STRING(255)
LOCATIONCOL_ADDRESS_STATE	STRING(255)
LOCATIONCOL_ADDRESS_STREET	STRING(255)
LOCATIONCOL_COORDINATES_LATITUDE	DOUBLE
LOCATIONCOL_COORDINATES_LONGITUDE	DOUBLE
LOCATIONCOL_DISPLAYNAME	STRING(255)
MODIFIED	DATETIMEOFFSET(26)
MULTILINECOL	STRING(63999)
NAME	STRING(2000)
NUMCOL	DOUBLE
PARENTREFERENCE_DRIVEID	STRING(2000)
PARENTREFERENCE_DRIVETYPE	STRING(2000)
PARENTREFERENCE_ID	STRING(2000)
PARENTREFERENCE_LISTID	STRING(2000)
PARENTREFERENCE_PATH	STRING(2000)
PARENTREFERENCE_SHAREPOINTIDS_LISTID	STRING(2000)
PARENTREFERENCE_SHAREPOINTIDS_LISTITEMID	STRING(2000)
PARENTREFERENCE_SHAREPOINTIDS_LISTITEMUNIQUEID	STRING(2000)
PARENTREFERENCE_SHAREPOINTIDS_SITEID	STRING(2000)
PARENTREFERENCE_SHAREPOINTIDS_SITEURL	STRING(2000)
PARENTREFERENCE_SHAREPOINTIDS_SYNCRESOURCEID	STRING(2000)

Column Name	Data Type
PARENTREFERENCE_SHAREPOINTIDS_TENANTDISPLAYNAME	STRING(2000)
PARENTREFERENCE_SHAREPOINTIDS_TENANTID	STRING(2000)
PARENTREFERENCE_SHAREPOINTIDS_WEBID	STRING(2000)
PARENTREFERENCE_SITEID	STRING(2000)
PERSONCOL	STRING(255)
PICTURECOL_DESCRIPTION	STRING(255)
PICTURECOL_URL	STRING(255)
SHAREPOINTIDS_LISTID	STRING(2000)
SHAREPOINTIDS_LISTITEMID	STRING(2000)
SHAREPOINTIDS_LISTITEMUNIQUEID	STRING(2000)
SHAREPOINTIDS_SITEID	STRING(2000)
SHAREPOINTIDS_SITEURL	STRING(2000)
SHAREPOINTIDS_SYNCRESOURCEID	STRING(2000)
SHAREPOINTIDS_TENANTDISPLAYNAME	STRING(2000)
SHAREPOINTIDS_TENANTID	STRING(2000)
SHAREPOINTIDS_WEBID	STRING(2000)
SINGLELINECOL	STRING(255)
TASKCOL	STRING(255)
TEXTCOL	STRING(63999)
TITLE	STRING(255)
WEBURL	STRING(2000)

DEFAULTCOLUMNSLIST

Columns

The DEFAULTCOLUMNSLIST table includes all the columns a user can specify in a list. The driver exposes all user-specified columns by default. The column names indicate the type of column created. The ID column marked with an asterisk is the primary key.

Column Name	Data Type
ID*	INT32
BOOLEANCOL	BOOLEAN
CHOICECOL	STRING(255)
CURRENCYCOL	DECIMAL(19, 4)
LINKCOL_DESCRIPTION	STRING(255)
LINKCOL_URL	STRING(255)
LOCATIONCOL_ADDRESS_CITY	STRING(255)
LOCATIONCOL_ADDRESS_COUNTRYORREGION	STRING(255)
LOCATIONCOL_ADDRESS_POSTALCODE	STRING(255)
LOCATIONCOL_ADDRESS_STATE	STRING(255)
LOCATIONCOL_ADDRESS_STREET	STRING(255)
LOCATIONCOL_COORDINATES_LATITUDE	DOUBLE
LOCATIONCOL_COORDINATES_LONGITUDE	DOUBLE
LOCATIONCOL_DISPLAYNAME	STRING(255)
MULTILINECOL	STRING(63999)
NUMCOL	DOUBLE
PERSONCOL	STRING(255)
PICTURECOL_DESCRIPTION	STRING(255)
PICTURECOL_URL	STRING(255)
SINGLELINECOL	STRING(255)
TASKCOL	STRING(255)

Column Name	Data Type
TEXTCOL	STRING(63999)
TITLE	STRING(255)

