Progress DataDirect

Progress DataDirect for ODBC for HubSpot

User's Guide

Release 8.0.0

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for HubSpot Driver

The Progress® DataDirect® for ODBC™ for HubSpot™ driver supports SQL read-only access for ODBC applications to HubSpot. To support SQL access to HubSpot, the driver creates a relational map of the HubSpot data model and translates SQL statements to HubSpot requests. In addition, the driver employs a SQL engine component that provides support to SQL constructs unavailable in HubSpot. 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 HubSpot Data Model on page 107.

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?

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

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

- Release Notes: https://www.progress.com/odbc/release-history/
- DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf

Highlights of 8.0.0 Release

- Supports SQL read-only access to HubSpot.
- 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.
- Supports HubSpot data types through data type inference. See Data types on page 22 for details.
- Supports OAuth 2.0 authentication. See OAuth 2.0 authentication on page 48 for supported grant types and further details.
- Supports API key authentication. See API key authentication on page 47 for details.
- Supports the handling of large result sets with paging and the Fetch Size (FetchSize) connection option. See Fetch Size on page 68 for details.
- On Windows platforms, includes a redesigned ODBC 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 37 and Generating connection strings with the Configuration Manager on page 39 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 46 for details.

Driver requirements

Data source and platform requirements

Refer to Supported Configurations for the latest data source and platform requirements.

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.
- An application that is compatible with components that were built using Microsoft Visual Studio 2015 compiler and the standard Win32 threading model.
- 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.
- An application that is compatible with components that were built using Microsoft C/C++ Optimizing Compiler Version 14.00.40310.41 and the standard Windows 64 threading model.
- 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**: To configure a new file data source, click **Add** to display a list of installed drivers. Select your driver and click **Advanced** to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.
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`.

For API key authentication:

- **Authentication Method**: Determines which authentication method the driver uses during the course of a session. Set to **14 - URLParameter**.
- **API Key**: Enter the API key required for authenticating to HubSpot.

Note: See API key authentication on page 47 for details.

For OAuth 2.0 access token flow:

- **Authentication Method**: Determines which authentication method the driver uses during the course of a session. Set to **24 - OAuth2**.
- **Access Token**: Enter the access token required to authenticate to HubSpot.

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

For OAuth 2.0 refresh token grant:

- **Authentication Method**: Determines which authentication method the driver uses during the course of a session. Set to **24 - OAuth2**.
- **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 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.

  Important: The refresh 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 48 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 on page 19 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 on page 57 provides a complete list of supported properties by functionality.

- Configuring data sources with the Configuration Manager on page 37 guides you through using the GUI to configure the driver.
- Performance considerations on page 49 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 the test was successful, the window displays a confirmation message.
9. 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.
10. 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 on page 83: 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 39

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:

c) Set the ODBCINI environment variable to point to the path from the root directory to the system information file where your data source resides. For example, if you use an installation directory of `/opt/odbc` and the default system information file name, you would enter:

- **Korn or Bourne shell**: `ODBCINI=/opt/odbc/odbc.ini; export ODBCINI`
- **C shell**: `setenv ODBCINI /opt/odbc/odbc.ini`

d) Set the library path environment variable for your Linux operating system to include the directory containing your JVM's `libjvm.so` file, and that directory's parent directory. The Library Path Environment Variable on Linux is `LD_LIBRARY_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 properties.

API key authentication

```
[ODBC Data Sources]
HubSpot=DataDirect 8.0 HubSpot

[HubSpot]
Driver=ODBCHOME/lib/xxhubspot28.yy
...
AuthenticationMethod=14
...
API Key=api_key
...
```

OAuth 2.0 access token flow

```
[ODBC Data Sources]
HubSpot=DataDirect 8.0 HubSpot

[HubSpot]
Driver=ODBCHOME/lib/xxhubspot28.yy
...
AuthenticationMethod=24
...
AccessToken=access_token
...
```

OAuth 2.0 refresh token grant

```
[ODBC Data Sources]
HubSpot=DataDirect 8.0 HubSpot

[HubSpot]
Driver=ODBCHOME/lib/xxhubspot28.yy
...
AuthenticationMethod=24
...
ClientID=client_id
...
ClientSecret=client_secret
...
RefreshToken=refresh_token
...
```

See Configuration through the system information (odbc.ini) file on page 40 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 39, DSN-less connections, for more information. For examples, see Connection string examples on page 19.

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 on page 19 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 suites your environment.

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

   - Connection option descriptions on page 57 provides a complete list of supported options by functionality.
   - Performance considerations on page 49 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 on page 83: 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 19

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 39

API key authentication

This string includes the connection options used to connect with API key authentication.

```
DRIVER=DataDirect 8.0 HubSpot;AuthenticationMethod=14;
APIKey=api_key;[attribute=value[;...]];
```

where:

api_key

specifies the API key when authenticating with API key authentication.

attribute=value

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

The following example connection string includes the options for connecting with the API key authentication.

```
DRIVER=DataDirect 8.0 HubSpot;AuthenticationMethod=14;
APIKey=1234ab5cd-1a2b-3a4b-678uvw-xyz91011;
```

See also

Using a connection string on page 39
API key authentication on page 47
Connection option descriptions on page 57

OAuth 2.0 access token flow

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

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string."

Note: On Windows platforms, as opposed to using a third-party application such as Postman, you can use the Progress DataDirect HubSpot Configuration Manager to obtain an access token to support the access token flow. See "Obtaining access and refresh tokens using the Configuration Manger" for details. Alternatively, you can obtain an access token from HubSpot. Refer to the HubSpot documentation at https://help.hubspot.com/ 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.

```
DRIVER=DataDirect 8.0 HubSpot;AccessToken=access_token;[attribute=value[;...]];
```

where:

access_token

specifies the access token required to authenticate to HubSpot. This option allows you to set the access token manually.

attribute=value

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

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

```
DRIVER=DataDirect 8.0 HubSpot;AccessToken=123=abCD/EfGh4Ijk+L;
```

See also

OAuth 2.0 authentication on page 48
Connection option descriptions on page 57

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 Token URI connection option.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string."

Note: On Windows platforms, as opposed to using a third-party application such as Postman, you can use the Progress DataDirect HubSpot Configuration Manager to obtain a refresh token to support the refresh token grant. See "Obtaining access and refresh tokens using the Configuration Manger" for details. Alternatively, you can obtain a refresh token from HubSpot. Refer to the HubSpot documentation at https://help.hubspot.com for details.

```
DRIVER=DataDirect 8.0 HubSpot;ClientID=client_id;ClientSecret=client_secret;
RefreshToken=refresh_token;[attribute=value[;...]];
```

where:

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.

refresh_token

specifies the refresh token for your application when authenticating with OAuth 2.0.

attribute=value

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

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

```
DRIVER=DataDirect 8.0 HubSpot;ClientID=29453d6f-6789-25gh-gd8g-44tk3c527831;
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

See also

OAuth 2.0 authentication on page 48
Connection option descriptions on page 57

Proxy server

This string includes the connection options you may need to connect through a proxy server with OAuth 2.0 access token flow.

```
DRIVER=DataDirect 8.0 HubSpot;AccessToken=access_token;
ProxyHost=proxy_host;ProxyPassword=proxy_password;
ProxyPort=proxy_port;ProxyUser=proxy_user;[attribute=value[;...]];
```

where:

access_token

specifies the access token used to authenticate to the HubSpot 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.

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.

attribute=value

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

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

```
DRIVER=DataDirect 8.0 HubSpot;AccessToken=123=abCD/EfGh4Ijk+L;
ProxyHost=pserver;ProxyPassword=secret;ProxyPort=808;ProxyUser=jsmith;
```

See also

Connection option descriptions on page 57

Data types

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

Table 1: HubSpot Data Types

HubSpot Data Type	ODBC Data Type
BigInt	SQL_BIGINT
Binary	SQL_BINARY
Bit	SQL_BIT

HubSpot Data Type	ODBC Data Type
Boolean	SQL_BIT
Char	SQL_CHAR
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Double	SQL_DOUBLE
Float	SQL_FLOAT
GUID	SQL_CHAR
Integer	SQL_INTEGER
JSON	SQL_VARCHAR
LongVarBinary	SQL_LONGVARBINARY
LongVarChar	SQL_LONGVARCHAR
NVarChar	SQL_UNICODE_VARCHAR
SmallInt	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
TinyInt	SQL_TINYINT
VarBinary	SQL_VARBINARY
VarChar	SQL_VARCHAR

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.

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 JDBC Drivers Reference* or use the links below to view some common topics:

- "JDBC support" describes support for JDBC interfaces and methods for the Progress DataDirect for JDBC drivers.
- "JDBC extensions" describes the JDBC extensions provided by the com.ddtek.jdbc.extensions package.
- "SQL escape sequences for JDBC" provides an overview of SQL escape sequences for JDBC. In addition, it documents the scalar functions that you use in SQL statements.
- "Security best practices for JDBC applications" describes the security best practices you should employ when developing and deploying your application with the driver.

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 HubSpot, Version 0001

2

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 to connect to a data source and execute SQL.

1. After you have configured the data source, navigate to the `instal_dir\samples\example` directory.
2. Open the application:
   - On Windows, double-click the `Example.exe` file.
   - On Linux, run the example application.

   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.SYSTEM_TABLES
   ```

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

Note: Refer to the `example.txt` file in the `example` subdirectory for a detailed explanation of how to build and use this application.

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, `DataDirectHubSpot.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 HubSpot Connector**.
5. Click **Connect**. Then, from the **Progress DataDirect HubSpot Connector** window, provide the following information. Then, click **OK**.
   - **Data Source:** Enter a name for the data source. For example, `HubSpot 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 HubSpot.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.

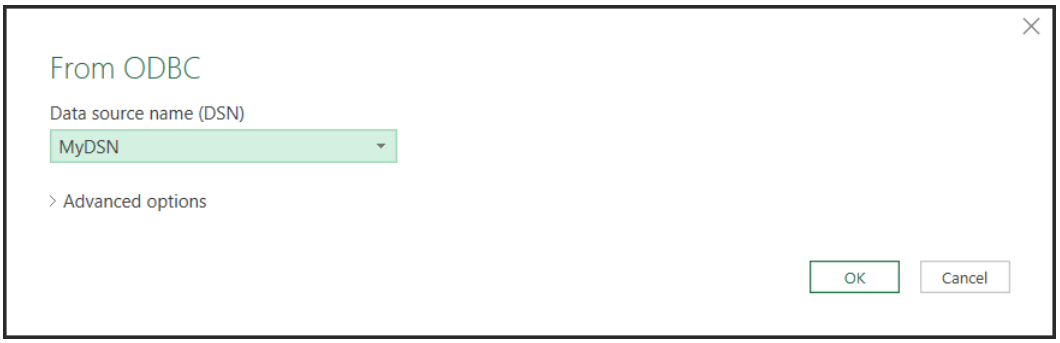
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

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.

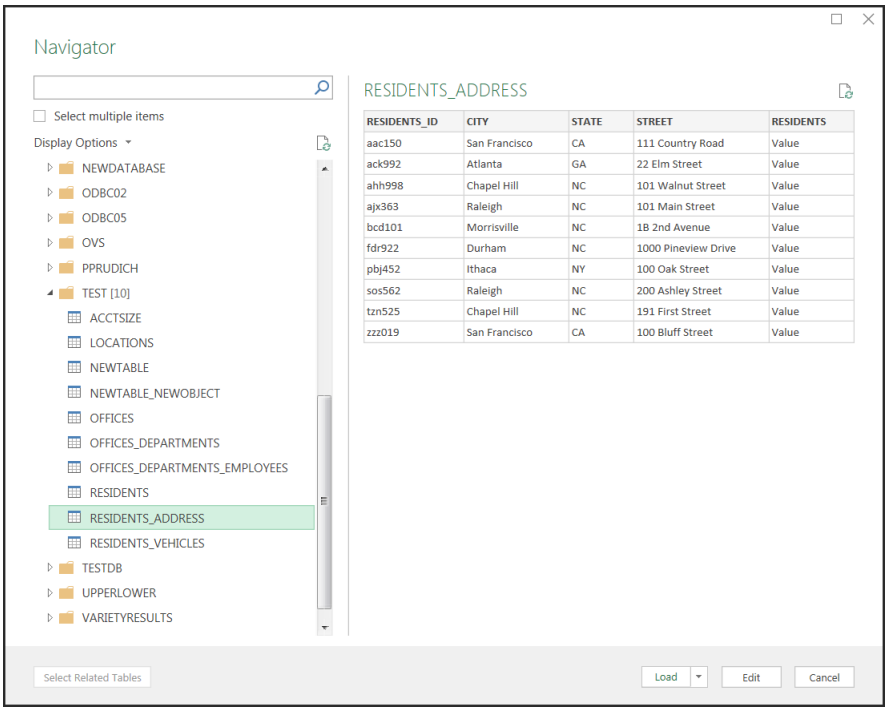


Select your data source from the Data Source Name (DSN) drop down; then, click **OK**.

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.

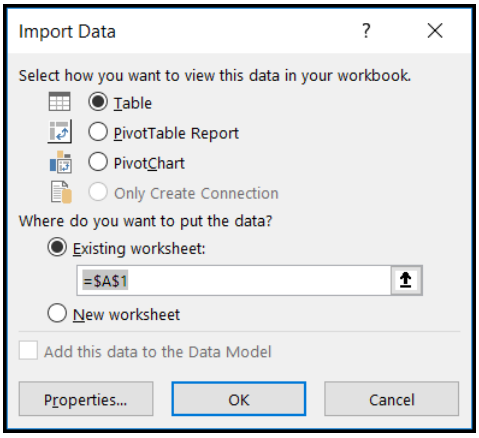


From the list, select the tables you want to access. A preview of your data will appear in the pane on the right. Optionally, click **Edit** to modify the results using the Query Editor. Refer to the Microsoft Excel product documentation for detailed information on using the Query Editor.

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/.

3

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)
- 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.

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 43

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 40

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/ivhubspotxx.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, Progress DataDirect HubSpot 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 a driver Setup dialog box, 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 HubSpot Configuration Manager by selecting ODBC Administrator from the Progress DataDirect program group.
2. Select a 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 dialog.

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

- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  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 driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The HubSpot 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.

   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

Configuration through the system information (odbc.ini) file on page 40
Generating connection strings with the Configuration Manager on page 39
Connection string examples on page 19

Generating connection strings with the Configuration Manager

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

The Progress DataDirect HubSpot 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

Testing connections and queries with the Configuration Manager on page 45
Configuring data sources with the Configuration Manager on page 37
Testing connections and queries with the Configuration Manager on page 45

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 driver for Linux or Windows is:

```
DSN=HubSpot;AccessToken=123=abCD/EfGh4Ijk+L
```

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

```
FILEDSN=HubSpot.dsn;AccessToken=123=abCD/EfGh4Ijk+L
```

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

```
DRIVER=DataDirect 8.0 HubSpot;AccessToken=123=abCD/EfGh4Ijk+L;
```

See also

Connection option descriptions on page 57
Connection string examples on page 19

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:

```
HubSpot=DataDirect 8.0 HubSpot Driver
```

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, `[HubSpot]`. 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]
IANAAppCodePage=4
InstallDir=/opt/odbc
Trace=0
TraceFile=odbctrace.out
TraceDll=/opt/odbc/lib/ivtrc28.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10
```

The IANAAppCodePage keyword defines the default value that the Linux driver uses if individual data sources have not specified a different value. See "IANAAppCodePage" in the "Connection option descriptions" for details.

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

DSN-less connections on page 43
Connection option descriptions on page 57
Sample odbcinst.ini file on page 44
File data sources on page 44

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]
HubSpot=DataDirect 8.0 HubSpot Driver

[HubSpot]
Driver=ODBCHOME/lib/ivhubspot28.so
Description=DataDirect 8.0 HubSpot Driver
AccessToken=
APIKey=
ApplicationUsingThreads=1
AuthenticationMethod=24
AuthURI=
ClientID=
ClientSecret=
DebugRecord=
FetchSize=100
JVMArgs=
JVMClasspath=
JVMPath=
LogConfigFile=ddlogging.properties
ProxyHost=
ProxyPassword=
ProxyPort=0
ProxyUser=
RefreshToken=
RedirectURI=
ReportCodepageConversionErrors=0
Scope=
SQLEngineMode=2
TokenURI=https://api.hubapi.com/oauth/v1/token
WSRetryCount=5
WSTimeout=120

[ODBC]
IANAAppCodePage=4
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.

See also

Connection option descriptions on page 57

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 HubSpot=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

ODBCINST on page 35
Configuration through the system information (odbc.ini) file on page 40

Sample odbcinst.ini file

The following is a sample odbcinst.ini. All occurrences of ODBCHOME 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 HubSpot=Installed

[DataDirect 8.0 HubSpot]
Driver=ODBCHOME/lib/ivhubspot28.so
JarFile=ODBCHOME/java/lib/hubspot.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.

IANAAppCodePage=4
InstallDir=ODBCHOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBCHOME/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 HubSpot
AccessToken=123=abCD/EfGh4Ijk+L
AuthenticationMethod=24
```

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\HubSpot.dsn
```

or, on Linux:

```
FILEDSN=/home/users/john/filedsn/HubSpot.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/HubSpot.dsn;
AccessToken=123=abCD/EfGh4Ijk+L
AuthenticationMethod=24;
```

See also

Configuring and connecting to data sources on page 33
DSN-less connections on page 43
Configuration through the system information (odbc.ini) file on page 40

Testing connections and queries with the Configuration Manager

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

To test your connection and query:

1. Open the HubSpot Configuration Manager by selecting the ODBC Administrator from the Progress DataDirect group.

   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 HubSpot 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 37

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.

Authentication

The driver supports the following authentication methods:

- *API key authentication* authenticates using an API key.
- *OAuth 2.0 authentication* authenticates using the OAuth 2.0 authentication flows.

By default, the driver is configured to use OAuth 2.0 authentication (`AuthenticationMethod=24`).

See also

Authentication Method on page 63

API key authentication

The driver supports API key authentication. It allows you to authenticate to the REST endpoints using an API key. To learn how to generate an API key for your account, refer to the HubSpot documentation.

To configure the driver to use API key authentication:

- Set the Authentication Method (AuthenticationMethod) option to `14`.
- Set the API Key (APIKey) option to specify the API key used to authenticate the client application to the HubSpot API.

The following examples show the connection information required to establish a session using API key authentication.

Connection string

```
DRIVER=DataDirect 8.0 HubSpot;AuthenticationMethod=14;
APIKey=1234ab5cd-1a2b-3a4b-678uvw-xyz91011;
```

odbc.ini

```
[HubSpot]
Driver=ODBCHOME/lib/ivhubspot28.so
Description=DataDirect 8.0 HubSpot
...
AuthenticationMethod=14
```

```
...
APIKey=1234ab5cd-1a2b-3a4b-678uvw-xyz91011
...
```

OAuth 2.0 authentication

The driver supports the following OAuth 2.0 grant types for accessing web instances of HubSpot services:

- Access token flow on page 48
- Refresh token grant on page 48

Access token flow

The access token authentication flow passes the access token directly from the client to the HubSpot service for authentication. Unlike other grant types, authentication credentials, such as authorization codes, are not exchanged in return for the access code. Instead, the access token is obtained from sources external to the flow and specified using the Access Token option.

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 Access Token (AccessToken) option 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 string

```
DRIVER=DataDirect 8.0 HubSpot;
AccessToken=123=abCD/EfGh4Ijk+L;
```

odbc.ini file

```
[HubSpot]
Driver=ODBCHOME/lib/ivhubspot28.so
Description=DataDirect 8.0 HubSpot
...
AccessToken=123=abCD/EfGh4Ijk+L
...
```

See also

Connection option descriptions on page 57

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 Token URI connection option.

To configure the driver to use an access token flow:

- 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 key for your application.

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

- Set the Refresh Token (RefreshToken) option 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.

- Optionally, set the Scope (Scope) option to specify a space-separated list of OAuth scopes to limit the permissions granted by the access token.

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

Connection string

```
DRIVER=DataDirect 8.0 HubSpot;
ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831
```

odbc.ini file

```
[HubSpot]
Driver=ODBCHOME/lib/ivhubspot28.so
Description=DataDirect 8.0 HubSpot
...
ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
...
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
...
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831
...
```

See also

Connection option descriptions on page 57

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.

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.

See also

Connection option descriptions on page 57

4

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.

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: `19962`

     64-bit: `19961`
   - **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\hubspot.jar}`

     Linux: `{.:/home/user1/`*`install_dir`*`/java/lib/hubspot.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 Java SE 8 or higher 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 80
JVM Classpath on page 70
JVM Path on page 71
JVM Arguments on page 69
Server Proxy Host on page 77
Server Port Number on page 77
Server Proxy User on page 79
Server Proxy Password on page 78

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

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.phoenix.sql.Server
       -port <port_number> -Dhttp.proxyHost=<proxy_host> -Dhttp.proxPort=<proxy_port>

        -Dhttp.proxyUser=<proxy_user> -Dhttp.proxyPassword=<proxy_password>
```

For example:

```
java -Xmx1024m -cp "/opt/Progress/DataDirect/ODBC_80_64bit/java/lib/hubspot.jar"
        com.ddtek.phoenix.sql.Server -port 19961 -Dhttp.proxyHost=myhost@mydomain.com
        -Dhttp.proxPort=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: *`install_dir`*`/java/lib/hubspot.jar`
`-port`	Specifies a valid port on which the SQL engine listens for requests from the driver. We recommend specifying one of the following values: • `19962` (32-bit drivers) • `19961` (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*.

5

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.

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

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

OAuth 2.0 options

The following table summarizes options used for OAuth 2.0 authentication.

Table 4: OAuth 2.0 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
RedirectURI (o2ru)	No default value
Refresh Token on page 75	No default value
Scope (oas)	No default value
TokenURI (o2tu)	`https://api.hubapi.com/oauth/v1/token`

API key authentication options

The following table summarizes options used for API key authentication.

Table 5: API key authentication options

Attribute (Short Name)	Default
AuthenticationMethod (am)	24 (OAuth)
APIKey	No default value

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)	*`install_dir`*`\java\lib\hubspot.jar`
JVMPath (jvmp)	*`install_dir`*`\jre\bin\java.exe`
ServerPortNumber (spn)	32-bit: `19962` 64-bit: `19961`
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

Attribute (Short Name)	Default
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 timeout.

Table 8: Web service options

Attribute (Short Name)	Default
WSRetryCount (wsrc)	`5`
WSTimeout (wst)	`120`

Additional options

The following table summarizes additional connection options.

Table 9: Additional Options

Attribute (Short Name)	Default
ApplicationUsingThreads (aut)	`1` (true)
DebugRecord (dbgrecord)	No default value
ExtendedOptions (xo)	No default value
FetchSize (fs)	`100` (rows)
LogConfigFile (lgcf)	`ddlogging.properties`
ReportCodepageConversionErrors (rcce)	`0`

For details, see the following topics:

- Access Token
- API Key
- Application Using Threads

- Authentication Method
- Authorization URI
- Client ID
- Client Secret
- Data Source Name
- Debug Folder
- Description
- Extended Options
- Fetch Size
- JVM Arguments
- JVM Classpath
- JVM Path
- Log Config File
- Proxy Host
- Proxy Password
- Proxy Port
- Proxy User
- Redirect URI
- Refresh Token
- Report Codepage Conversion Errors
- Scope
- Server Port Number
- Server Proxy Host
- Server Proxy Password
- Server Proxy Port
- Server Proxy User
- SQL Engine Mode
- SQL Service
- Token URI
- Web Service Retry Count
- Web Service Timeout

Access Token

Attribute

AccessToken (atok)

Purpose

Specifies the access token used to authenticate to HubSpot 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 48

API Key

Attribute

APIKey

Purpose

Specifies the API key used to authenticate the client application to the HubSpot API when API key authentication is enabled (`AuthenticationMethod=UrlParameter`). To learn how to generate an API key for your account, refer to the HubSpot documentation.

Valid Values

string

where:

string

is the API key you have generated for your account.

Default Value

No default value

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 49

Authentication Method

Attribute

AuthenticationMethod (am)

Purpose

Determines which authentication mechanism the driver uses when establishing a connection.

Valid Values

24 | 14

Behavior

If set to 14 (URLParameter), the driver uses an API key to authenticate to the REST endpoints. See "API key authentication" for details.

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

Default Value

24 (OAuth2)

See also

Authentication on page 47

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 48

Client ID

Attribute

ClientID (cid)

Purpose

Specifies the client ID key for your application when authenticating to HubSpot 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

OAuth 2.0 authentication on page 48

Client Secret

Attribute

ClientSecret (clse)

Purpose

Specifies the client secret for your application when authenticating to HubSpot 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

OAuth 2.0 authentication on page 48

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

Debug Folder

Attribute

DebugRecord (dbgrecord)

Purpose

Specifies the directory where the driver generates debug record files. When a value is specified, the driver records server requests and responses to a set of files stored in this location. These files assist in troubleshooting by providing a method for Technical Support to reproduce and debug issues for REST services that are not publicly accessible.

Important: Debug record files may capture security-related headers, such as auth or token headers. Before sending Technical Support debug files, review the content to remove any confidential information that may have been recorded.

Valid Values

debug_record_folder

where:

debug_record_folder

is the location of the folder where the debug record files are to be generated. For example, `C:\Temp\MyDebug Folder`.

Note: To specify this value using the GUI, you must click the **Record** box . Next, click **Select**; then, browse to or create the file location.

Notes

- The specified directory must exist.
- You must have write access to the specified directory.
- The contents of the specified directory are deleted every time a connection is established.
- For more information, refer to "Enabling debug record mode" in the *Progress DataDirect for ODBC Drivers Reference*.
- For assistance, contact Technical Support.

Default Value

No default value

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

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 49

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) or 3 (Broker):

```
-Xmx256m
```

For all other configurations:

```
-Xmx1024m
```

See also

SQL Engine Mode on page 80
Windows environment variables on page 34

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 HubSpot and MongoDB drivers on Windows, you would specify a value of {c:\install_dir\java\lib\hubspot.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\hubspot.jar}
```

On Linux:

```
{.:/home/user1/install_dir/java/lib/hubspot.jar}
```

Default Value

No default value

See also

Using the SQL engine server on page 51

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

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

See also

Proxy server on page 21

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

See also

Proxy server on page 21

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 72

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

See also

Proxy server on page 21

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 48

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 48

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`

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

See also

OAuth 2.0 authentication on page 48

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: 19962

64-bit: 19961

See also

Using the SQL engine server on page 51

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 51

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 51

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 51

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 51

SQL Engine Mode

Attribute

SQLEngineMode (sem)

Purpose

Specifies whether the driver's SQL engine runs in the same 32-bit 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 51

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 51

Token URI

Attribute

TokenURI (o2tu)

Purpose

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

Valid Values

String

where:

String

is the endpoint used to retrieve OAuth 2.0 access tokens. For example, `https://example.com/oauth2/authorize/`.

Default Value

`https://api.hubapi.com/oauth/v1/token`

See also

OAuth 2.0 authentication on page 48

Web Service Retry Count

Attribute

WSRetryCount (wsrc)

Purpose

Specifies the number of times the driver retries a timed-out Select request. The timeout period is specified by the Web Service Timeout (WSTimeout) option.

Valid Values

`0` | *x*

where:

x

is a positive integer

Behavior

If set to `0`, the driver does not retry timed-out requests after the initial unsuccessful attempt.

If set to *x*, the driver retries the timed-out request the specified number of times.

Default Value

`5`

Web Service Timeout

Attribute

WSTimeout (wst)

Purpose

Specifies the time, in seconds, that the driver waits for a response to a web service request.

Valid Values

`0` | *x*

where:

x

is a positive integer that defines the number of seconds the driver waits for a response to a web service request.

Behavior

If set to `0`, the driver waits indefinitely for a response; there is no timeout.

If set to *x*, the driver uses the value as the default timeout, measured in seconds, for any statement created by the connection.

If a Select request times out and Web Service Retry Count (WSRetryCount) is set to retry timed-out requests, the driver retries the request the specified number of times.

Default Value

`120`

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)
- Explain Plan
- Select
- 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: `ALTER SESSION SET CURRENT_SCHEMA=`
Stmt_Call_Limit	Local	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: `ALTER SESSION SET STMT_CALL_LIMIT=150`

Ws_Call_Count	Remote	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:
		`ALTER SESSION SET .WS_CALL_COUNT=0`
		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:
		`SELECT * from information_schema.system_remote_sessions WHERE session_id = cursessionid()`

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.

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 97

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

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 95

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 97
Subqueries on page 95

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 97
Subqueries on page 95

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 97
Subqueries on page 95

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 97

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
```

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	*n* where *n* is any valid integer value in the range of the INTEGER data type	`12 or -34 or 0`

SQL Type	Literal Syntax	Example
BOOLEAN	Min Value: 0 Max Value: 1	`0` `1`
DATE	DATE'*date*'	`'2010-05-21'`
DATETIME	TIMESTAMP'*ts*'	`'2010-05-21 18:33:05.025'`

DECIMAL	*n.f*
	where:
	n
	is the integral part
	f
	is the fractional part
		`0.25`
		`3.1415`
		`-7.48`

DOUBLE	*n.f*E*x* 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	'*hex_value*'	`'000482ff'`
LONGVARCHAR	'*value*'	`'This is a string literal'`
TIME	TIME'*time*'	`'2010-05-21 18:33:05.025'`
VARCHAR	'*value*'	`'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 quotion 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`
><	"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`

Operator	Purpose	Example
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.	`SELECT * FROM emp WHERE NOT (job IS NULL)` `SELECT * FROM emp WHERE NOT (sal BETWEEN 1000 AND 2000)`
AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise, returns UNKNOWN.	`SELECT * FROM emp WHERE job = 'CLERK' AND deptno = 10`
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE; otherwise, returns UNKNOWN.	`SELECT * FROM emp WHERE job = 'CLERK' OR deptno = 10`

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
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]`
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`

7

Introduction to the HubSpot Data Model

The HubSpot data model is defined using a collection of standard JSON documents that contain the data, identifiers, and object relationships for a given service. These documents are stored on URL endpoints that are accessible using sets of proprietary REST API calls. To expose HubSpot resources to SQL applications, the driver maps HubSpot endpoints to a set of relational parent and child tables. The following sections describe the relational tables exposed by the driver along with their corresponding HubSpot REST API call.

For details, see the following topics:

- CAMPAIGNS
- COMMENTS
- COMPANIES
- COMPANIES_V3
- CONTACT
- CONTACTIDENTITYPROFILEIDENTITIES
- CONTACTIDENTITYPROFILES
- CONTACTLISTIDENTITYPROFILEIDENTITIES
- CONTACTLISTIDENTITYPROFILES
- CONTACTLISTS
- CONTACTS
- CONTACTSINLIST
- CONTACTS_V3
- CRMASSOCIATIONS
- DEALASSOCIATEDCOMPANYIDS
- DEALASSOCIATEDCONTACTIDS
- DEALASSOCIATEDDEALIDS
- DEALASSOCIATEDTICKETIDS
- DEALASSOCIATIONS
- DEALPIPELINES
- DEALPIPELINESTAGES
- DEALS
- DEALS_V3
- DOMAINS
- ECOMMERCESYNCERRORS
- EMAILSUBSCRIPTIONS
- ENGAGEMENTASSOCIATEDCOMPANIES
- ENGAGEMENTASSOCIATEDCONTACTIDS
- ENGAGEMENTASSOCIATEDCONTENTIDS
- ENGAGEMENTASSOCIATEDDEALIDS
- ENGAGEMENTASSOCIATEDOWNERIDS
- ENGAGEMENTASSOCIATEDQUOTEIDS
- ENGAGEMENTASSOCIATEDTICKETIDS

- ENGAGEMENTASSOCIATEDWORKFLOWIDS
- ENGAGEMENTS
- ENGAGEMENTSSCHEDULEDTASKS
- EVENTS
- FILES
- FOLDERS
- FORMFIELDGROUPS
- FORMFIELDOPTIONS
- FORMFIELDS
- FORMFIELDSELECTEDOPTIONS
- FORMS
- LINEITEMS
- LINEITEMS_V3
- MARKETINGEMAILS
- OWNERS
- OWNERSREMOTELISTS
- PAGES
- PRODUCTS
- PRODUCTS_V3
- QUOTES_V3
- SOCIALMEDIACHANNELS
- SOCIALMEDIAMESSAGES
- TASKS
- TEMPLATES
- TICKETPIPELINES
- TICKETPIPELINESTAGES
- TICKETS
- TICKETS_V3
- URLMAPPINGS
- WORKFLOW

ALLEMAILCAMPAIGNIDS

Endpoint

```
https://api.hubapi.com/marketing-emails/v1/emails
```

Parent Table

MARKETINGEMAILS

Columns

The ALLEMAILCAMPAIGNIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
MARKETINGEMAILS_ID*	BigInt	References: MARKETINGEMAILS.ID
POSITION*	Integer	
ALLEMAILCAMPAIGNID	Integer	

ANALYTICS

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{BreakdownBy}/{TimePeriod}
```

Columns

The ANALYTICS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
BREAKDOWNBY*	VarChar(64)
TIMEPERIOD*	VarChar(64)
STARTDATE*	VarChar(64)
ENDDATE*	VarChar(64)
BOUNCERATE	Double
CONTACTS	Integer
CONTACTSPERPAGEVIEW	Double

Column Name	Data Type
CONTACTTOCUSTOMERRATE	Double
CUSTOMERS	Integer
LEADS	Integer
LEADSPERVIEW	Double
MARKETINGQUALIFIEDLEADS	Integer
NEWVISITORSESSIONRATE	Double
OPPORTUNITIES	Integer
PAGEVIEWSMINUSEXITS	Integer
PAGEVIEWSPERSESSION	Double
RAWVIEWS	Integer
RETURNINGVISITS	Integer
SALESQUALIFIEDLEADS	Integer
SESSIONTOCONTACTRATE	Double
STANDARDVIEWS	Integer
SUBSCRIBERS	Integer
TIMEPERSESSION	Double
TOTAL	Integer
VISITORS	Integer
VISITS	Integer

ANALYTICSBREAKDOWNS

Endpoint

`https://api.hubapi.com/analytics/v2/reports/{BreakdownBy}/{TimePeriod}`

Parent Table

ANALYTICS

Columns

The ANALYTICSBREAKDOWNS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ANALYTICS_BREAKDOWNBY*	VarChar(64)	References: ANALYTICS.BREAKDOWNBY
ANALYTICS_TIMEPERIOD*	VarChar(64)	References: ANALYTICS.TIMEPERIOD
ANALYTICS_STARTDATE*	VarChar(64)	References: ANALYTICS.STARTDATE
ANALYTICS_ENDDATE*	VarChar(64)	References: ANALYTICS.ENDDATE
POSITION*	Integer	
BOUNCERATE	Double	
BREAKDOWN	VarChar(64)	
CONTACTS	Integer	
CONTACTSPERPAGEVIEW	Double	
CONTACTTOCUSTOMERRATE	Double	
CUSTOMERS	Integer	
LEADS	Integer	
LEADSPERVIEW	Double	
MAPPEDBREAKDOWN	Integer	
MARKETINGQUALIFIEDLEADS	Integer	
NEWVISITORSESSIONRATE	Double	
OPPORTUNITIES	Integer	
PAGEVIEWSMINUSEXITS	Integer	
PAGEVIEWSPERSESSION	Double	
RAWVIEWS	Integer	
RETURNINGVISITS	Integer	
SALESQUALIFIEDLEADS	Integer	
SESSIONTOCONTACTRATE	Double	
STANDARDVIEWS	Integer	

Column Name	Data Type	Notes
SUBSCRIBERS	Integer	
TIMEPERSESSION	Double	
VISITORS	Integer	
VISITS	Integer	

ANALYTICSCONTENTBREAKDOWNS

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{ContentType}/{TimePeriod}
```

Parent Table

ANALYTICSCONTENTS

Columns

The ANALYTICSCONTENTBREAKDOWNS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ANALYTICSCONTENTS_CONTENTTYPE*	VarChar(64)	References: ANALYTICSCONTENTS .CONTENTTYPE
ANALYTICSCONTENTS_TIMEPERIOD*	VarChar(64)	References: ANALYTICSCONTENTS.TIMEPERIOD
ANALYTICSCONTENTS_STARTDATE*	VarChar(64)	References: ANALYTICSCONTENTS.STARTDATE
ANALYTICSCONTENTS_ENDDATE*	VarChar(64)	References: ANALYTICSCONTENTS.ENDDATE
POSITION*	Integer	
BREAKDOWN	BigInt	
CONTACTS	Integer	
CTAVIEWS	Integer	
ENTRANCES	Integer	
EXITS	Integer	
EXITSPERPAGEVIEW	Double	
LEADS	Integer	

Column Name	Data Type	Notes
MAPPEDBREAKDOWN	Integer	
PAGEBOUNCERATE	Double	
PAGETIME	Integer	
PAGEVIEWSMINUSEXITS	Integer	
RAWVIEWS	Integer	
STANDARDVIEWS	Integer	
SUBMISSIONS	Integer	
TIMEPERPAGEVIEW	Double	

ANALYTICSCONTENTS

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{ContentType}/{TimePeriod}
```

Columns

The ANALYTICSCONTENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
CONTENTTYPE*	VarChar(64)
TIMEPERIOD*	VarChar(64)
STARTDATE*	VarChar(64)
ENDDATE*	VarChar(64)
CONTACTS	Integer
CTAVIEWS	Integer
ENTRANCES	Integer
EXITS	Integer
EXITSPERPAGEVIEW	Double
LEADS	Integer

Column Name	Data Type
PAGEBOUNCERATE	Double
PAGETIME	Integer
PAGEVIEWSMINUSEXITS	Integer
RAWVIEWS	Integer
STANDARDVIEWS	Integer
SUBMISSIONS	Integer
TIMEPERPAGEVIEW	Double
TOTAL	Integer

ANALYTICSSPECIFICOBJECT

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{ObjectType}/{TimePeriod}
```

Columns

The ANALYTICSSPECIFICOBJECT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTTYPE*	VarChar(64)
TIMEPERIOD*	VarChar(64)
STARTDATE*	VarChar(64)
ENDDATE*	VarChar(64)
CTAVIEWS	Integer
ENTRANCES	Integer
EXITS	Integer
EXITSPERPAGEVIEW	Double
FORMVIEWS	Integer
INTERACTIONS	Integer

Column Name	Data Type
PAGEBOUNCERATE	Double
RAWVIEWS	Integer
TIMEPERPAGEVIEW	Double
TOTAL	Integer
VISIBLES	Integer

ANALYTICSSPECIFICOBJECTBREAKDOWNS

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{ObjectType}/{TimePeriod}
```

Parent Table

ANALYTICSSPECIFICOBJECT

Columns

The ANALYTICSSPECIFICOBJECTBREAKDOWNS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ANALYTICSSPECIFICOBJECT_OBJECTTYPE*	VarChar(64)	References: ANALYTICSSPECIFICOBJECT .OBJECTTYPE
ANALYTICSSPECIFICOBJECT_TIMEPERIOD*	VarChar(64)	References: ANALYTICSSPECIFICOBJECT .TIMEPERIOD
ANALYTICSSPECIFICOBJECT_STARTDATE*	VarChar(64)	References: ANALYTICSSPECIFICOBJECT .STARTDATE
ANALYTICSSPECIFICOBJECT_ENDDATE*	VarChar(64)	References: ANALYTICSSPECIFICOBJECT .ENDDATE
POSITION*	Integer	
BREAKDOWN	GUID	
CTAVIEWS	Integer	
ENTRANCES	Integer	
EXITS	Integer	

Column Name	Data Type	Notes
EXITSPERPAGEVIEW	Double	
FORMVIEWS	Integer	
INTERACTIONS	Integer	
MAPPEDBREAKDOWN	Integer	
PAGEBOUNCERATE	Double	
RAWVIEWS	Integer	
TIMEPERPAGEVIEW	Double	
VISIBLES	Integer	

BLOGAUTHORS

Endpoint

`https://api.hubapi.com/blogs/v3/blog-authors`

Columns

The BLOGAUTHORS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CREATED	Timestamp(3)
DELETEDAT	Timestamp(3)
DISPLAYNAME	VarChar(64)
EMAIL	VarChar(64)
FACEBOOK	VarChar(64)
FULLNAME	VarChar(64)
HASSOCIALPROFILES	Boolean
LINKEDIN	VarChar(64)
PORTALID	Integer

Column Name	Data Type
SLUG	VarChar(64)
TWITTER	VarChar(64)
UPDATED	Timestamp(3)
USERID	Integer
USERNAME	VarChar(64)
WEBSITE	VarChar(64)

BLOGPOSTS

Endpoint

`https://api.hubapi.com/content/api/v2/blog-posts`

Columns

The BLOGPOSTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
AUTHORNAME	VarChar(64)
AUTHORUSERID	Integer
BLOGAUTHORID	BigInt
CONTENTGROUPID	BigInt
CREATEDAT	Timestamp(3)
DELETEDAT	Timestamp(3)
NAME	VarChar(64)
POSTSUMMARY	VarChar(64)
PUBLISHEDAT	Timestamp(3)
SLUG	VarChar(64)

Column Name	Data Type
UPDATEDAT	Timestamp(3)
URL	VarChar(94)

BLOGS

Endpoint

`https://api.hubapi.com/content/api/v2/blogs`

Columns

The BLOGS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
ALLOWCOMMENTS	Boolean
COMMENTSHOULDCREATECONTACT	Boolean
CREATEDAT	Timestamp(3)
HTMLTITLE	VarChar(64)
LANGUAGE	VarChar(64)
NAME	VarChar(64)
POSTSPERLISTINGPAGE	Integer
POSTSPERRSSFEED	Integer
PUBLICTITLE	VarChar(64)
ROOTURL	VarChar(76)
SHOWSOCIALLINKFACEBOOK	Boolean
SHOWSOCIALLINKLINKEDIN	Boolean
SHOWSOCIALLINKTWITTER	Boolean
SLUG	VarChar(64)
UPDATEDAT	Timestamp(3)

BLOGTOPICIDS

Endpoint

`https://api.hubapi.com/content/api/v2/blog-posts`

Parent Table

BLOGPOSTS

Columns

The BLOGTOPICIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
BLOGPOSTS_ID*	BigInt	References: BLOGPOSTS.ID
POSITION*	Integer	
BLOGTOPICID	BigInt	

BLOGTOPICS

Endpoint

`https://api.hubapi.com/blogs/v3/topics`

Columns

The BLOGTOPICS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CREATED	Timestamp(3)
DELETEDAT	Timestamp(3)
DESCRIPTION	VarChar(64)
NAME	VarChar(64)
PORTALID	Integer

Column Name	Data Type
SLUG	VarChar(64)
UPDATED	Timestamp(3)

CAMPAIGNINFO

Endpoint

```
https://api.hubapi.com/email/public/v1/campaigns/{CampaignId}
```

Columns

The CAMPAIGNINFO table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
CAMPAIGNID*	Integer
APPID	Integer
APPNAME	VarChar(64)
CONTENTID	BigInt
DELIVERED	Integer
ID	Integer
NAME	VarChar(128)
NUMINCLUDED	Integer
NUMQUEUED	Integer
OPEN	Integer
PROCESSED	Integer
SENT	Integer
SUBJECT	VarChar(64)
SUBTYPE	VarChar(64)
TYPE	VarChar(64)

CAMPAIGNS

Endpoint

```
https://api.hubapi.com/email/public/v1/campaigns/by-id
```

Columns

The CAMPAIGNS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
APPID	Integer
APPNAME	VarChar(64)

COMMENTS

Endpoint

```
https://api.hubapi.com/comments/v3/comments
```

Columns

The COMMENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
COLLECTIONID	BigInt
COMMENT	VarChar(64)
CONTENTAUTHOREMAIL	VarChar(64)
CONTENTAUTHORNAME	VarChar(64)
CONTENTCREATEDAT	BigInt
CONTENTID	BigInt
CONTENTPERMALINK	VarChar(94)
CONTENTTITLE	VarChar(64)

Column Name	Data Type
CREATEDAT	Timestamp(3)
DELETEDAT	Timestamp(3)
EXTRACONTEXT	VarChar(64)
FIRSTNAME	VarChar(64)
LASTNAME	VarChar(64)
LEGACYID	Integer
PARENTID	Integer
PORTALID	Integer
POSTID	Integer
REPLYINGTO	VarChar(64)
STATE	VarChar(64)
THREADID	VarChar(73)
USERAGENT	VarChar(64)
USEREMAIL	VarChar(64)
USERIP	VarChar(64)
USERNAME	VarChar(64)
USERREFERRER	VarChar(64)
USERURL	VarChar(64)

COMPANIES

Endpoint

```
https://api.hubapi.com/companies/v2/companies/{CompanyId}
```

Columns

The COMPANIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
COMPANYID*	BigInt
ALLOWNERIDS	Integer
ANALYTICSFIRSTTIMESTAMP	Date
ANALYTICSNOOFPAGEVIEWS	Integer
ANALYTICSSOURCE	VarChar(64)
ANALYTICSSOURCEDATA1	VarChar(64)
ANALYTICSSOURCEDATA2	VarChar(64)
ANNUALREVENUE	BigInt
CITY	VarChar(64)
CLOSEDATE	Date
DATECREATED	Date
DAYSTOCLOSE	Integer
DESCRIPTION	VarChar(64)
DOMAIN	VarChar(64)
FILEMANAGERKEY	VarChar(82)
FIRSTCONTACTCREATEDATE	Date
FIRSTDEALCREATEDDATE	Date
HUBSPOTOWNERASSIGNDATE	Date
HUBSPOTOWNERID	Integer
INDUSTRY	VarChar(64)
ISDELETED	Boolean
LASTCONTACTED	Date
LASTMODIFIEDDATE	Date
LASTUPDATED	Date
LIFECYCLESTAGE	VarChar(64)
LINKEDPAGELINK	VarChar(64)

Column Name	Data Type
NAME	VarChar(64)
NOOFANALYTICSVISITS	Integer
NOOFASSOCIATEDCONTACTS	Integer
NOOFASSOCIATEDDEALS	Integer
NOOFCHILDREN	Integer
NOOFCONTACTEDNOTES	Integer
NOOFEMPLOYEES	Integer
NOOFNOTES	Integer
NOTESNEXTACTIVITYDATE	Date
OBJECTID	BigInt
PARENTCOMPANYID	BigInt
PHONENO	VarChar(64)
PORTALID	Integer
PREDICTIVECONTACTSSCORE	Double
RECENTDEALAMOUNT	Integer
RECENTDEALCLOSEDATE	Date
STATE	VarChar(64)
TIMEZONE	VarChar(64)
TOTALREVENUE	Integer
TYPE	VarChar(64)
WEBSITE	VarChar(64)
YEARFOUNDED	Integer
ZIP	VarChar(64)

COMPANIES_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/companies
```

Columns

The COMPANIES_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
ARCHIVED	Boolean
ASSOCIATEDCONTACTS	VarChar(64)
ASSOCIATEDDEALS	VarChar(64)
CONTACTEDNOTES	VarChar(64)
CREATEDAT	Timestamp(9)
FACEBOOKCOMPANYPAGE	VarChar(82)
FIRSTCONTACTCREATEDATE	VarChar(64)
FIRSTDEALCREATEDDATE	VarChar(64)
HSANALYTICSFIRSTTIMESTAMP	VarChar(64)
HSANALYTICSNUMPAGEVIEWS	VarChar(64)
HSANALYTICSNUMVISITS	VarChar(64)
HSANALYTICSSOURCE	VarChar(64)
HSANALYTICSSOURCEDATA1	VarChar(64)
HSANALYTICSSOURCEDATA2	VarChar(64)
HSPREDICTIVECONTACTSCOREV2	VarChar(64)
ISPUBLIC	VarChar(64)
LASTCONTACTED	VarChar(64)
LINKEDINCOMPANYPAGE	VarChar(123)

Column Name	Data Type
NOTES	VarChar(64)
PROPERTIES_ADDRESS	VarChar(64)
PROPERTIES_ANNUALREVENUE	Integer
PROPERTIES_CITY	VarChar(64)
PROPERTIES_COUNTRY	VarChar(64)
PROPERTIES_DESCRIPTION	VarChar(423)
PROPERTIES_DOMAIN	VarChar(64)
PROPERTIES_INDUSTRY	VarChar(64)
PROPERTIES_LIFECYCLESTAGE	VarChar(64)
PROPERTIES_LINKEDINBIO	VarChar(423)
PROPERTIES_NAME	VarChar(64)
PROPERTIES_NOTES_LAST_UPDATED	VarChar(64)
PROPERTIES_NOTES_NEXT_ACTIVITY_DATE	VarChar(64)
PROPERTIES_NUMBEROFEMPLOYEES	Integer
PROPERTIES_PHONE	VarChar(64)
PROPERTIES_STATE	VarChar(64)
PROPERTIES_TIMEZONE	VarChar(64)
PROPERTIES_TWITTERHANDLE	VarChar(64)
PROPERTIES_WEB_TECHNOLOGIES	VarChar(417)
PROPERTIES_WEBSITE	VarChar(64)
PROPERTIES_ZIP	VarChar(64)
UPDATEDAT	Timestamp(9)
YEARFOUNDED	Integer

CONTACT

Endpoint

```
https://api.hubapi.com/contacts/v1/contact/vid/{vid}/profile
```

Columns

The CONTACT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
VID*	Integer
ADDRESS	VarChar(64)
CALCULATEDPHONENO	VarChar(64)
CALCULATEDPHONENOCOUNTRYCODE	VarChar(64)
CITY	VarChar(64)
DATECREATED	Timestamp(3)
EMAIL	VarChar(64)
FIRSTNAME	VarChar(64)
HSANALYTICSAVERAGEPAGEVIEWS	Integer
HSANALYTICSFIRSTTIMESTAMP	BigInt
HSANALYTICSFIRSTTOUCHCONVERTINGCAMPAIGN	VarChar(64)
HSANALYTICSFIRSTVISITTIMESTAMP	VarChar(64)
HSANALYTICSLASTTIMESTAMP	VarChar(64)
HSANALYTICSLASTTOUCHCONVERTINGCAMPAIGN	VarChar(64)
HSANALYTICSLASTVISITTIMESTAMP	VarChar(64)
HSANALYTICSNUMEVENTCOMPLETIONS	Integer
HSANALYTICSNUMPAGEVIEWS	Integer
HSANALYTICSNUMVISITS	Integer
HSANALYTICSREVENUE	Double

Column Name	Data Type
HSANALYTICSSOURCE	VarChar(64)
HSANALYTICSSOURCEDATA1	VarChar(64)
HSANALYTICSSOURCEDATA2	Integer
HSEMAILDOMAIN	VarChar(64)
HSEMAILOPTOUT	VarChar(64)
HSEMAILOPTOUT6841412	VarChar(64)
HSEMAILOPTOUT6841413	VarChar(64)
HSEMAILOPTOUT6841782	VarChar(64)
HSEMAILOPTOUT6841830	VarChar(64)
HSEMAILQUARANTINED	VarChar(64)
HSISCONTACT	VarChar(64)
HSLIFECYCLESTAGESUBSCRIBERDATE	Date
HSOBJECTID	Integer
HSPREDICTIVECONTACTSCOREV2	Double
HSPREDICTIVESCORINGTIER	VarChar(64)
HSSEARCHABLECALCULATEDPHONENUMBER	BigInt
HSSOCIALFACEBOOKCLICKS	Integer
HSSOCIALGOOGLEPLUSCLICKS	Integer
HSSOCIALLASTENGAGEMENT	VarChar(64)
HSSOCIALLINKEDINCLICKS	Integer
HSSOCIALNUMBROADCASTCLICKS	VarChar(64)
HSSOCIALTWITTERCLICKS	Integer
ISCONTACT	Boolean
LASTMODIFIEDDATE	Timestamp(3)
LASTNAME	VarChar(64)
LIFECYCLESTAGE	VarChar(64)

Column Name	Data Type
NUMCONVERSIONEVENTS	Integer
NUMUNIQUECONVERSIONEVENTS	Integer
PHONENO	VarChar(64)
PORTALID	Integer
PROFILEURL	VarChar(78)
PROPERTIES_HS_SOCIAL_NUM_BROADCAST_CLICKS_VALUE	Integer
STATE	VarChar(64)
ZIP	Integer

CONTACTIDENTITYPROFILEIDENTITIES

Endpoint

`https://api.hubapi.com/contacts/v1/lists/all/contacts/all`

Parent Table

CONTACTIDENTITYPROFILES

Columns

The CONTACTIDENTITYPROFILEIDENTITIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
CONTACTS_VID*	Integer	References: CONTACTS.VID
CONTACTIDENTITYPROFILES_POSITION*	Integer	References: CONTACTIDENTITYPROFILES .POSITION
POSITION*	Integer	
ISPRIMARY	Boolean	
TIMESTAMP	Timestamp(3)	
TYPE	VarChar(13)	
VALUE	VarChar(64)	

CONTACTIDENTITYPROFILES

Endpoint

`https://api.hubapi.com/contacts/v1/lists/all/contacts/all`

Parent Table

CONTACTS

Columns

The CONTACTIDENTITYPROFILES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
CONTACTS_VID*	Integer	References: CONTACTS.VID
POSITION*	Integer	
CONTACTVID	Integer	
DELETEDCHANGEDTIMESTAMP	Integer	
SAVEDAT	Timestamp(3)	

CONTACTLISTIDENTITYPROFILEIDENTITIES

Endpoint

`https://api.hubapi.com/contacts/v1/lists/{ListId}/contacts/all`

Parent Table

CONTACTLISTIDENTITYPROFILES

Columns

The CONTACTLISTIDENTITYPROFILEIDENTITIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
CONTACTSINLIST_LISTID*	Integer	References: CONTACTSINLIST.LISTID
CONTACTLISTIDENTITYPROFILES _POSITION*	Integer	References: CONTACTLISTIDENTITYPROFILES .POSITION
POSITION*	Integer	

Column Name	Data Type	Notes
ISPRIMARY	Boolean	
TIMESTAMP	BigInt	
TYPE	VarChar(13)	
VALUE	VarChar(64)	

CONTACTLISTIDENTITYPROFILES

Endpoint

`https://api.hubapi.com/contacts/v1/lists/{ListId}/contacts/all`

Parent Table

CONTACTSINLIST

Columns

The CONTACTLISTIDENTITYPROFILES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
CONTACTSINLIST_LISTID*	Integer	References: CONTACTSINLIST.LISTID
POSITION*	Integer	
DELETEDCHANGEDTIMESTAMP	Integer	
SAVEDATTIMESTAMP	BigInt	
VID	Integer	

CONTACTLISTS

Endpoint

`https://api.hubapi.com/contacts/v1/lists`

Columns

The CONTACTLISTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer
CREATEDAT	Timestamp(3)
FILTERSAGGREGATE	JSON(16777215)
INTERNALLISTID	Integer
ISDYNAMIC	Boolean
LASTPROCESSINGSTATECHANGEAT	Timestamp(3)
LASTSIZECHANGEAT	Timestamp(3)
LISTID	Integer
LISTSIZE	Integer
NAME	VarChar(64)
PROCESSINGSTATE	VarChar(64)
UPDATEDAT	Timestamp(3)

CONTACTS

Endpoint

```
https://api.hubapi.com/contacts/v1/lists/all/contacts/all
```

Columns

The CONTACTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
VID*	Integer
ADDEDAT	Timestamp(3)
COMPANY	VarChar(64)
FIRSTNAME	VarChar(64)
ISCONTACT	Boolean
LASTMODFIFIEDAT	Timestamp(3)

Column Name	Data Type
LASTNAME	VarChar(64)
PORTALID	Integer
PROFILEURL	VarChar(78)

CONTACTSINLIST

Endpoint

```
https://api.hubapi.com/contacts/v1/lists/{ListId}/contacts/all
```

Columns

The CONTACTSINLIST table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
LISTID*	Integer
ADDEDAT	Timestamp(3)
FIRSTNAME	VarChar(64)
ISCONTACT	Boolean
LASTMODFIFIEDAT	Timestamp(3)
LASTNAME	VarChar(64)
PROFILEURL	VarChar(78)
VID	Integer

CONTACTS_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/contacts
```

Columns

The CONTACTS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(9)
HSANALYTICSAVERAGEPAGEVIEWS	Integer
HSANALYTICSFIRSTTIMESTAMP	Timestamp(9)
HSANALYTICSFIRSTTOUCHCONVERTINGCAMPAIGN	VarChar(64)
HSANALYTICSFIRSTVISITTIMESTAMP	VarChar(64)
HSANALYTICSLASTTIMESTAMP	VarChar(64)
HSANALYTICSLASTTOUCHCONVERTINGCAMPAIGN	VarChar(64)
HSANALYTICSLASTVISITTIMESTAMP	VarChar(64)
HSANALYTICSNUMEVENTCOMPLETIONS	Integer
HSANALYTICSNUMPAGEVIEWS	Integer
HSANALYTICSNUMVISITS	Integer
HSANALYTICSREVENUE	Double
HSANALYTICSSOURCE	VarChar(64)
HSANALYTICSSOURCEDATA1	VarChar(64)
HSANALYTICSSOURCEDATA2	VarChar(64)
HSCALCULATEDPHONENUMBER	VarChar(64)
HSCALCULATEDPHONENUMBERCOUNTRYCODE	VarChar(64)
HSEMAILDOMAIN	VarChar(64)
HSEMAILOPTOUT	VarChar(64)
HSEMAILOPTOUT6841412	VarChar(64)
HSEMAILOPTOUT6841413	VarChar(64)
HSEMAILOPTOUT6841782	VarChar(64)
HSEMAILOPTOUT6841830	VarChar(64)
HSEMAILQUARANTINED	VarChar(64)

Column Name	Data Type
HSEMAILRECIPIENTFATIGUERECOVERYTIME	VarChar(64)
HSISCONTACT	VarChar(64)
HSLIFECYCLESTAGESUBSCRIBERDATE	Timestamp(9)
HSPREDICTIVECONTACTSCOREV2	Double
HSPREDICTIVESCORINGTIER	VarChar(64)
HSSEARCHABLECALCULATEDPHONENUMBER	BigInt
HSSOCIALFACEBOOKCLICKS	Integer
HSSOCIALGOOGLEPLUSCLICKS	Integer
HSSOCIALLASTENGAGEMENT	VarChar(64)
HSSOCIALLINKEDINCLICKS	Integer
HSSOCIALNUMBROADCASTCLICKS	Integer
HSSOCIALTWITTERCLICKS	Integer
NUMCONVERSIONEVENTS	Integer
NUMUNIQUECONVERSIONEVENTS	Integer
PROPERTIES_ADDRESS	VarChar(64)
PROPERTIES_ASSOCIATEDCOMPANYID	VarChar(64)
PROPERTIES_CITY	VarChar(64)
PROPERTIES_EMAIL	VarChar(64)
PROPERTIES_FIRSTNAME	VarChar(64)
PROPERTIES_LASTNAME	VarChar(64)
PROPERTIES_LIFECYCLESTAGE	VarChar(64)
PROPERTIES_PHONE	VarChar(64)
PROPERTIES_STATE	VarChar(64)
PROPERTIES_ZIP	Integer
UPDATEDAT	Timestamp(9)

CRMASSOCIATIONS

Endpoint

```
https://api.hubapi.com/crm-associations/v1/associations/{ObjectId}/HUBSPOT_DEFINED/{DefinitionId}
```

Columns

The CRMASSOCIATIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTID*	BigInt
DEFINITIONID*	Integer
RESULTID	Integer

DEALASSOCIATEDCOMPANYIDS

Endpoint

```
https://api.hubapi.com/deals/v1/deal/{DealId}
```

Parent Table

DEALS

Columns

The DEALASSOCIATEDCOMPANYIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALS_DEALID*	Integer	References: DEALS.DEALID
POSITION*	Integer	
DEALASSOCIATEDCOMPANYID	BigInt	

DEALASSOCIATEDCONTACTIDS

Endpoint

`https://api.hubapi.com/deals/v1/deal/{DealId}`

Parent Table

DEALS

Columns

The DEALASSOCIATEDCONTACTIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALS_DEALID*	Integer	References: DEALS.DEALID
POSITION*	Integer	
DEALASSOCIATEDCONTACTID	BigInt	

DEALASSOCIATEDDEALIDS

Endpoint

`https://api.hubapi.com/deals/v1/deal/{DealId}`

Parent Table

DEALS

Columns

The DEALASSOCIATEDDEALIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALS_DEALID*	Integer	References: DEALS.DEALID
POSITION*	Integer	
DEALASSOCIATEDDEALID	BigInt	

DEALASSOCIATEDTICKETIDS

Endpoint

`https://api.hubapi.com/deals/v1/deal/{DealId}`

Parent Table

DEALS

Columns

The DEALASSOCIATEDTICKETIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALS_DEALID*	Integer	References: DEALS.DEALID
POSITION*	Integer	
DEALASSOCIATEDTICKETID	BigInt	

DEALASSOCIATIONS

Endpoint

`https://api.hubapi.com/deals/v1/deal/associated/{ObjectType}/{ObjectId}/paged`

Columns

The DEALASSOCIATIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTTYPE*	VarChar(64)
DEALID	Integer
OBJECTID	BigInt
PORTALID	Integer

DEALPIPELINES

Endpoint

```
https://api.hubapi.com/crm-pipelines/v1/pipelines/deals
```

Columns

The DEALPIPELINES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PIPELINEID*	VarChar(64)
ACTIVE	Boolean
CREATEDAT	Integer
DEFAULT_	Boolean
DISPLAYORDER	Integer
LABEL	VarChar(64)
OBJECTTYPE	VarChar(64)
OBJECTTYPEID	VarChar(64)
UPDATEDAT	VarChar(64)

DEALPIPELINESTAGES

Endpoint

```
https://api.hubapi.com/crm-pipelines/v1/pipelines/deals
```

Parent Table

DEALPIPELINES

Columns

The DEALPIPELINESTAGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALPIPELINES_PIPELINEID*	VarChar(64)	References: DEALPIPELINES.PIPELINEID

Column Name	Data Type	Notes
POSITION*	Integer	
ACTIVE	Boolean	
CREATEDAT	Integer	
DISPLAYORDER	Integer	
ISCLOSED	Boolean	
PROBABILITY	Double	
STAGEID	VarChar(64)	
STAGENAME	VarChar(64)	
UPDATEDAT	VarChar(64)	

DEALS

Endpoint

`https://api.hubapi.com/deals/v1/deal/{DealId}`

Columns

The DEALS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
DEALID*	Integer
ALLACCESSIBLETEAMIDS	Integer
ALLOWNERIDS	Integer
ALLTEAMIDS	Integer
AMOUNT	Double
AMOUNTINHOMECURRENCY	Double
CLOSEDATE	Date
CLOSEDDEALAMOUNT	Integer
CLOSEDDEALAMOUNTINHOMECURRENCY	Integer

Column Name	Data Type
CREATEDATE	Date
DAYSTOCLOSE	Integer
DEALNAME	VarChar(64)
DEALSTAGEPROBABILITY	Integer
HUBSPOTCREATEDATE	Date
HUBSPOTOBJECTID	Integer
HUBSPOTOWNERID	Integer
HUBSPOTTEAMID	Integer
ISDELETED	Boolean
LASTMODIFIEDDATE	Date
NUMBEROFCONTACTS	Integer
OWNERASSIGNEDDATE	Date
PORTALID	Integer
PROJECTEDDEALAMOUNT	Double
PROJECTEDDEALAMOUNTINHOMECURRENCY	Double

DEALS_V3

Endpoint

`https://api.hubapi.com/crm/v3/objects/deals`

Columns

The DEALS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
AMOUNTINHOMECURRENCY	Double
ARCHIVED	Boolean

Column Name	Data Type
CREATEDAT	Timestamp(9)
DAYSTOCLOSE	Integer
HSALLACCESSIBLETEAMIDS	Integer
HSALLOWNERIDS	Integer
HSALLTEAMIDS	Integer
HSCLOSEDAMOUNT	Integer
HSCLOSEDAMOUNTINHOMECURRENCY	Integer
HSCREATEDATE	Timestamp(9)
HSDEALSTAGEPROBABILITY	Integer
HSPROJECTEDAMOUNT	Double
HSPROJECTEDAMOUNTINHOMECURRENCY	Double
HUBSPOTOWNERASSIGNEDDATE	Timestamp(9)
HUBSPOTOWNERID	Integer
HUBSPOTTEAMID	Integer
NUMASSOCIATEDCONTACTS	Integer
PROPERTIES_AMOUNT	Double
PROPERTIES_CLOSEDATE	Timestamp(0)
PROPERTIES_DEALNAME	VarChar(64)
UPDATEDAT	Timestamp(9)

DOMAINS

Endpoint

`https://api.hubapi.com/cos-domains/v1/domains`

Columns

The DOMAINS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CREATED	Timestamp(3)
DOMAIN	VarChar(64)
ISANYPRIMARY	Boolean
ISDNSCORRECT	Boolean
ISLEGACYDOMAIN	Boolean
ISRESOLVING	Boolean
MANUALLYMARKEDASRESOLVING	Boolean
PRIMARYBLOGPOST	Boolean
PRIMARYEMAIL	Boolean
PRIMARYLANDINGPAGE	Boolean
PRIMARYLEGACYPAGE	Boolean
PRIMARYSITEPAGE	Boolean
SECONDARYTODOMAIN	VarChar(64)
UPDATED	Timestamp(3)

ECOMMERCESYNCERRORS

Endpoint

`https://api.hubapi.com/extensions/ecomm/v2/sync/errors`

Columns

The ECOMMERCESYNCERRORS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
POSITION*	Integer
CHANGEDAT	Timestamp(3)
DETAILS	VarChar(256)

Column Name	Data Type
ERROREDAT	Timestamp(3)
EXTERNALOBJECTID	Integer
OBJECTTYPE	VarChar(64)
PORTALID	Integer
STATUS	VarChar(64)
STOREID	VarChar(64)
TYPE	VarChar(64)

EMAILSUBSCRIPTIONS

Endpoint

`https://api.hubapi.com/email/public/v1/subscriptions`

Columns

The EMAILSUBSCRIPTIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ACTIVE	Boolean
CATEGORY	VarChar(64)
CHANNEL	VarChar(64)
DESCRIPTION	VarChar(88)
INTERNAL	Boolean
INTERNALNAME	VarChar(64)
NAME	VarChar(64)
PORTALID	Integer

ENGAGEMENTASSOCIATEDCOMPANIES

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDCOMPANIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDCOMPANY	BigInt	

ENGAGEMENTASSOCIATEDCONTACTIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDCONTACTIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDCONTACTID	BigInt	

ENGAGEMENTASSOCIATEDCONTENTIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDCONTENTIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDCONTENTID	BigInt	

ENGAGEMENTASSOCIATEDDEALIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDDEALIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDDEALID	BigInt	

ENGAGEMENTASSOCIATEDOWNERIDS

Endpoint

```
https://api.hubapi.com/engagements/v1/engagements/paged
```

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDOWNERIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDOWNERID	BigInt	

ENGAGEMENTASSOCIATEDQUOTEIDS

Endpoint

```
https://api.hubapi.com/engagements/v1/engagements/paged
```

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDQUOTEIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDQUOTEID	BigInt	

ENGAGEMENTASSOCIATEDTICKETIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDTICKETIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDTICKETID	BigInt	

ENGAGEMENTASSOCIATEDWORKFLOWIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDWORKFLOWIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDWORKFLOWID	BigInt	

ENGAGEMENTS

Endpoint

```
https://api.hubapi.com/engagements/v1/engagements/paged
```

Columns

The ENGAGEMENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
ATTACHMENTS	JSON(16777215)
BODY	VarChar(64)
CREATEDAT	Timestamp(3)
CREATEDBY	Integer
DATETIME	Timestamp(3)
ENGAGEMENT_BODYPREVIEW	VarChar(64)
ISACTIVE	Boolean
MODIFIEDBY	Integer
OWNERID	Integer
PORTALID	Integer
STATUS	VarChar(64)
SUBJECT	VarChar(64)
TYPE	VarChar(64)
UPDATEDAT	Timestamp(3)

ENGAGEMENTSSCHEDULEDTASKS

Endpoint

```
https://api.hubapi.com/engagements/v1/engagements/paged
```

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTSSCHEDULEDTASKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
DATETIME	Timestamp(3)	
ENGAGEMENTID	BigInt	
ENGAGEMENTTYPE	VarChar(64)	
PORTALID	Integer	
TASKTYPE	VarChar(64)	
UUID	VarChar(64)	

EVENTS

Endpoint

`https://api.hubapi.com/reports/v2/events`

Columns

The EVENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	VarChar(64)
LABEL	VarChar(64)
NAME	VarChar(64)
STATUS	VarChar(64)

FILES

Endpoint

`https://api.hubapi.com/filemanager/api/v2/files`

Columns

The FILES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
ALTKEY	VarChar(130)
ALTURL	VarChar(172)
CREATEDAT	Timestamp(3)
DELETEDAT	Timestamp(3)
EXTENSION	VarChar(64)
FILE_HASH	VarChar(64)
FOLDERID	BigInt
FRIENDLYURL	VarChar(174)
HEIGHT	Integer
ICONFRIENDLYURL	VarChar(187)
ICONIMAGENAME	VarChar(64)
ICONS3URL	VarChar(186)
ISARCHIVED	Boolean
MEDIUMFRIENDLYURL	VarChar(190)
MEDIUMIMAGENAME	VarChar(64)
MEDIUMS3URL	VarChar(189)
META_ALLOWS_ANONYMOUS_ACCESS	Boolean
META_URL_SCHEME	VarChar(64)
NAME	VarChar(64)

Column Name	Data Type
PORTAL_ID	Integer
S3_URL	VarChar(172)
SIZE	Integer
THUMBFRIENDLYURL	VarChar(189)
THUMBIMAGENAME	VarChar(64)
THUMBS3URL	VarChar(187)
TITLE	VarChar(64)
TYPE	VarChar(64)
UPDATEDAT	Timestamp(3)
URL	VarChar(174)
WIDTH	Integer

FOLDERS

Endpoint

```
https://api.hubapi.com/filemanager/api/v2/folders
```

Columns

The FOLDERS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CATEGORY	Integer
CDNPURGEEMBARGOTIME	VarChar(64)
CREATEDAT	Timestamp(3)
DELETED	Boolean
DELETEDAT	Timestamp(3)
FULLPATH	VarChar(64)

Column Name	Data Type
HIDDEN	Boolean
NAME	VarChar(64)
PARENTFOLDERID	BigInt
PORTALID	Integer
UPDATEDAT	Timestamp(3)

FORMFIELDGROUPS

Endpoint

`https://api.hubapi.com/forms/v2/forms`

Parent Table

FORMS

Columns

The FORMFIELDGROUPS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
FORMS_PORTALID*	Integer	References: FORMS.PORTALID
POSITION*	Integer	
CONTENT	VarChar(64)	
DEFAULT_	Boolean	
ISSMARTGROUP	Boolean	
TYPE	VarChar(64)	

FORMFIELDOPTIONS

Endpoint

`https://api.hubapi.com/forms/v2/fields/{FormGUID}`

Parent Table

FORMFIELDS

Columns

The FORMFIELDOPTIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
FORMFIELDS_FORMGUID*	VarChar(64)	References: FORMFIELDS.FORMGUID
FORMFIELDS_LABEL*	VarChar(64)	References: FORMFIELDS.LABEL
POSITION*	Integer	
DISCRIPTION	VarChar(64)	
DISPLAYORDER	Integer	
DOUBLEDATA	Double	
ISHIDDEN	Boolean	
LABEL	VarChar(64)	
READONLY	Boolean	
VALUE	VarChar(64)	

FORMFIELDS

Endpoint

`https://api.hubapi.com/forms/v2/fields/{FormGUID}`

Columns

The FORMFIELDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
FORMGUID*	VarChar(64)
LABEL*	VarChar(64)
DEFAULTVALUE	VarChar(64)
DISPLAYORDER	Integer

Column Name	Data Type
ENABLED	Boolean
FIELDTYPE	VarChar(64)
GROUPNAME	VarChar(64)
HIDDEN	Boolean
ISSMARTFIELD	Boolean
NAME	VarChar(75)
REQUIRED	Boolean
TYPE	VarChar(64)
VALIDATION_VALIDATIONBLOCKEDEMAILADDRESSES	JSON(16777215)
VALIDATIONDATA	VarChar(64)
VALIDATIONMESSAGE	VarChar(64)
VALIDATIONNAME	VarChar(64)
VALIDATIONUSEDEFAULTBLOCKLIST	Boolean

FORMFIELDSELECTEDOPTIONS

Endpoint

`https://api.hubapi.com/forms/v2/fields/{FormGUID}`

Parent Table

FORMFIELDS

Columns

The FORMFIELDSELECTEDOPTIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
FORMFIELDS_FORMGUID*	VarChar(64)	References: FORMFIELDS.FORMGUID
FORMFIELDS_LABEL*	VarChar(64)	References: FORMFIELDS.LABEL

Column Name	Data Type	Notes
POSITION*	Integer	
FORMFIELDSELECTEDOPTION	VarChar(64)	

FORMS

Endpoint

`https://api.hubapi.com/forms/v2/forms`

Columns

The FORMS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer
ACTION	VarChar(64)
CAMPAIGNGUID	VarChar(64)
CREATEDAT	Timestamp(3)
CSSCLASS	VarChar(64)
DELETEDAT	Timestamp(3)
ENDATTIMESTAMP	Timestamp(3)
FOLLOWUPID	VarChar(64)
FORMTYPE	VarChar(64)
GUID	GUID
IGNORECURRENTVALUES	Boolean
INLINEMESSAGE	VarChar(64)
ISDELETABLE	Boolean
ISPUBLISHED	Boolean
LEADNURTURINGCAMPAIGNID	VarChar(64)
METADATA	JSON(16777215)

Column Name	Data Type
METHOD	VarChar(64)
MIGRATEDFROM	VarChar(64)
MULTIVARIATETEST_CONTROLID	VarChar(64)
MULTIVARIATETEST_FINISHED	Boolean
MULTIVARIATETEST_VARIANTS	JSON(16777215)
MULTIVARIATETEST_WINNINGVARIANTID	VarChar(64)
NAME	VarChar(88)
NOTIFYRECIPIENTS	VarChar(64)
PARENTID	Integer
PERFORMABLEHTML	VarChar(64)
PUBLISHAT	Timestamp(3)
PUBLISHEDAT	Timestamp(3)
REDIRECT	VarChar(64)
STARTATTIMESTAMP	Timestamp(3)
SUBMITTEXT	VarChar(64)
UNPUBLISHAT	Timestamp(3)
UPDATEDAT	Timestamp(3)

LINEITEMS

Endpoint

`https://api.hubapi.com/crm-objects/v1/objects/line_items/paged`

Columns

The LINEITEMS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTTYPE*	VarChar(64)

Column Name	Data Type
DESCRIPTION	VarChar(64)
HSPRODUCTID	Integer
ISDELETED	Boolean
NAME	VarChar(64)
OBJECTID	Integer
PORTALID	Integer
PRICE	Integer
QUANTITY	Integer

LINEITEMS_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/line_items
```

Columns

The LINEITEMS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(6)
DESCRIPTION	VarChar(64)
HSPRODUCTID	Integer
NAME	VarChar(64)
PRICE	Integer
QUANTITY	Integer
UPDATEDAT	Timestamp(6)

MARKETINGEMAILS

Endpoint

`https://api.hubapi.com/marketing-emails/v1/emails`

Columns

The MARKETINGEMAILS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
AB	Boolean
ABHOURSTOWAIT	Integer
ABSAMPLESIZEDEFAULT	VarChar(64)
ABSAMPLINGDEFAULT	VarChar(64)
ABSOLUTEURL	VarChar(120)
ABSUCCESSMETRIC	VarChar(64)
ABTESTPERCENTAGE	Integer
ABVARIATION	Boolean
ANALYTICSPAGEID	BigInt
ANALYTICSPAGETYPE	VarChar(64)
ARCHIVED	Boolean
AUTHOR	VarChar(64)
AUTHORAT	BigInt
AUTHORNAME	VarChar(64)
CAMPAIGN	GUID
CAMPAIGNNAME	VarChar(64)
CAMPAIGNUTM	VarChar(64)
CANSPAMSETTINGSID	BigInt

Column Name	Data Type
CATEGORYID	Integer
CONTENTTYPECATEGORY	Integer
CREATED	Timestamp(3)
CREATEDBYID	Integer
CREATEPAGE	Boolean
CURRENTLYPUBLISHED	Boolean
CURRENTSTATE	VarChar(64)
CUSTOMREPLYTO	VarChar(64)
CUSTOMREPLYTOENABLED	Boolean
DOMAIN	VarChar(64)
EMAILBODY	VarChar(130)
EMAILBODYPLAINTEXT	VarChar(64)
EMAILCAMPAIGNGROUPID	Integer
EMAILNOTE	VarChar(64)
EMAILTYPE	VarChar(64)
FEEDBACKSURVEYID	Integer
FREEZEDATE	BigInt
FROMNAME	VarChar(64)
HASCONTENTACCESSRULES	Boolean
HTMLTITLE	VarChar(64)
ISGRAYMAILSUPPRESSIONENABLED	Boolean
ISPUBLISHED	Boolean
LANGUAGE	VarChar(64)
LIVEDOMAIN	VarChar(64)
MAILINGLISTSEXCLUDED	JSON(16777215)
MAILINGLISTSINCLUDED	JSON(16777215)

Column Name	Data Type
MAXRSSENTRIES	Integer
METADESCRIPTION	VarChar(64)
NAME	VarChar(64)
PAGEREDIRECTED	Boolean
PORTALID	Integer
PREVIEWKEY	VarChar(64)
PROCESSINGSTATUS	VarChar(64)
PUBLISHDATE	Date
PUBLISHEDAT	Timestamp(3)
PUBLISHEDBYEMAIL	VarChar(64)
PUBLISHEDBYID	Integer
PUBLISHEDBYNAME	VarChar(64)
PUBLISHEDURL	VarChar(120)
PUBLISHIMMEDIATELY	Boolean
REPLYTO	VarChar(64)
RESOLVEDDOMAIN	VarChar(64)
RSSEMAILBYTEXT	VarChar(64)
RSSEMAILCLICKTHROUGHTEXT	VarChar(64)
RSSEMAILCOMMENTTEXT	VarChar(64)
RSSEMAILENTRYTEMPLATEENABLED	Boolean
RSSEMAILURL	VarChar(64)
SLUG	VarChar(78)
SUBCATEGORY	VarChar(64)
SUBJECT	VarChar(64)
SUBSCRIPTION	Integer
SUBSCRIPTIONNAME	VarChar(64)

Column Name	Data Type
TEAMPERMS	JSON(16777215)
TEMPLATEPATH	VarChar(87)
TRANSACTIONAL	Boolean
UNPUBLISHEDAT	Timestamp(3)
UPDATED	Timestamp(3)
UPDATEDBYID	Integer
URL	VarChar(120)
USERPERMS	JSON(16777215)
USERSSHEADLINEASSUBJECT	Boolean
VIDSEXCLUDED	JSON(16777215)
VIDSINCLUDED	JSON(16777215)
VISIBLETOALL	Boolean
WIDGETS	JSON(16777215)

OWNERS

Endpoint

```
https://api.hubapi.com/owners/v2/owners/
```

Columns

The OWNERS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer
ACTIVESALESFORCEID	VarChar(64)
ACTIVEUSERID	Integer
CREATEDAT	Timestamp(3)
EMAIL	VarChar(76)

Column Name	Data Type
FIRSTNAME	VarChar(64)
HASCONTACTSACCESS	Boolean
ISACTIVE	Boolean
LASTNAME	VarChar(64)
OWNERID	Integer
TYPE	VarChar(9)
UPDATEDAT	Timestamp(3)
USERIDINCLUDINGINACTIVE	Integer

OWNERSREMOTELISTS

Endpoint

`https://api.hubapi.com/owners/v2/owners/`

Parent Table

OWNERS

Columns

The OWNERSREMOTELISTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
OWNERS_PORTALID*	Integer	References: OWNERS.PORTALID
POSITION*	Integer	
ACTIVE	Boolean	
ID	Integer	
OWNERID	Integer	
PORTALID	Integer	
REMOTEID	Integer	
REMOTETYPE	VarChar(10)	

PAGES

Endpoint

```
https://api.hubapi.com/content/api/v2/pages
```

Columns

The PAGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CREATEDAT	Timestamp(3)
CURRENTLIVEDOMAIN	VarChar(64)
DELETEDAT	Integer
ISARCHIVED	Boolean
NAME	VarChar(79)
PUBLISHDATE	BigInt
SLUG	VarChar(64)
SUBCATEGORY	VarChar(64)
UPDATEDAT	BigInt
URL	VarChar(91)

PRODUCTS

Endpoint

```
https://api.hubapi.com/crm-objects/v1/objects/products/{ObjectId}
```

Columns

The PRODUCTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer

Column Name	Data Type
DESCRIPTION	VarChar(64)
ISDELETED	Boolean
NAME	VarChar(64)
OBJECTID	Integer
OBJECTTYPE	VarChar(64)
PRICE	Integer

PRODUCTS_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/products
```

Columns

The PRODUCTS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(9)
PROPERTIES_DESCRIPTION	VarChar(64)
PROPERTIES_NAME	VarChar(64)
PROPERTIES_PRICE	Integer
UPDATEDAT	Timestamp(9)

QUOTES_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/quotes
```

Columns

The QUOTES_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(6)
PROPERTIES_HS_EXPIRATION_DATE	Timestamp(9)
PROPERTIES_HS_PUBLIC_URL_KEY	VarChar(64)
PROPERTIES_HS_STATUS	VarChar(64)
PROPERTIES_HS_TITLE	VarChar(64)
UPDATEDAT	Timestamp(6)

SOCIALMEDIACHANNELS

Endpoint

```
https://api.hubapi.com/broadcast/v1/channels/setting/publish/current
```

Columns

The SOCIALMEDIACHANNELS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer
ACCOUNTGUID	GUID
ACCOUNTSLUG	VarChar(64)
ACCOUNTTYPE	VarChar(64)
CHANNELGUID	GUID
CHANNELID	Integer
CHANNELKEY	VarChar(64)
CHANNELSLUG	VarChar(64)

Column Name	Data Type
CREATEDAT	Timestamp(3)
ISACTIVE	Boolean
ISHIDDEN	Boolean
ISSHARED	Boolean
TYPE	VarChar(64)
UPDATEDAT	Timestamp(3)
USERNAME	VarChar(64)

SOCIALMEDIAMESSAGES

Endpoint

`https://api.hubapi.com/broadcast/v1/broadcasts`

Columns

The SOCIALMEDIAMESSAGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
CONTENTBODY*	VarChar(256)
CAMPAIGNGUID	GUID
CAMPAIGNNAME	VarChar(64)
CHANNEL	VarChar(64)
CHANNELGUID	GUID
CLICKS	Integer
CLIENTTAG	VarChar(64)
CONTENTORIGINALBODY	VarChar(256)
CONTENTPHOTOURL	VarChar(512)
CONTENTUNCOMPRESSEDLINKS	VarChar(256)
CREATEDAT	Timestamp(3)

Column Name	Data Type
CREATEDBY	Integer
FINISHEDAT	Timestamp(3)
FOREIGNID	Integer
GROUPGUID	GUID
ISFAILED	Boolean
ISPENDING	Boolean
ISPUBLISHED	Boolean
ISRETRY	Boolean
MESSAGE	VarChar(256)
REMOTECONTENTID	VarChar(64)
REMOTECONTENTTYPE	VarChar(64)
STATUS	VarChar(64)
TRIGGERAT	VarChar(64)
UPDATEDBY	Integer

TASKS

Endpoint

`https://api.hubapi.com/calendar/v1/events/task`

Columns

The TASKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
STARTDATE*	VarChar(64)
ENDDATE*	VarChar(64)
AVATARURL	VarChar(64)
CAMPAIGNGUID	GUID

Column Name	Data Type
CATEGORY	VarChar(64)
CATEGORYID	Integer
CONTENTGROUPID	Integer
CONTENTID	Integer
CREATEDBY	VarChar(64)
DESCRIPTION	VarChar(64)
EVENTDATE	BigInt
EVENTTYPE	VarChar(64)
ID	Integer
NAME	VarChar(64)
OWNERID	BigInt
PORTALID	Integer
PREVIEWKEY	VarChar(64)
RECURRING	Boolean
SOCIALDISPLAYNAME	VarChar(64)
SOCIALUSERNAME	VarChar(64)
STATE	VarChar(64)
TOPICIDS	JSON(16777215)
URL	VarChar(256)

TEMPLATES

Endpoint

`https://api.hubapi.com/content/api/v2/templates`

Columns

The TEMPLATES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CATEGORYID	Integer
CDNMINIFIEDURL	VarChar(255)
CDNURL	VarChar(249)
DELETEDAT	Timestamp(3)
FOLDER	VarChar(84)
GENERATEDFROMLAYOUTID	BigInt
ISAVAILABLEFORNEWCONTENT	Boolean
ISFROMLAYOUT	Boolean
ISREADONLY	Boolean
LABEL	VarChar(60)
PATH	VarChar(109)
SOURCE	LongVarChar(95280)
UPDATEDAT	Timestamp(3)

TICKETPIPELINES

Endpoint

`https://api.hubapi.com/crm-pipelines/v1/pipelines/tickets`

Columns

The TICKETPIPELINES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PIPELINEID*	Integer
ACTIVE	Boolean
CREATEDAT	Integer
DEFAULT_	Boolean

Column Name	Data Type
DISPLAYORDER	Integer
LABEL	VarChar(64)
OBJECTTYPE	VarChar(64)
OBJECTTYPEID	VarChar(64)
UPDATEDAT	VarChar(64)

TICKETPIPELINESTAGES

Endpoint

`https://api.hubapi.com/crm-pipelines/v1/pipelines/tickets`

Parent Table

TICKETPIPELINES

Columns

The TICKETPIPELINESTAGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
TICKETPIPELINES_PIPELINEID*	Integer	References: TICKETPIPELINES.PIPELINEID
POSITION*	Integer	
ACTIVE	Boolean	
CREATEDAT	Integer	
DISPLAYORDER	Integer	
ISCLOSED	Boolean	
PROBABILITY	Double	
STAGEID	Integer	
STAGENAME	VarChar(64)	
UPDATEDAT	VarChar(64)	

TICKETS

Endpoint

```
https://api.hubapi.com/crm-objects/v1/objects/tickets/paged
```

Columns

The TICKETS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTTYPE*	VarChar(64)
CONTENT	VarChar(64)
CREATEDBY	VarChar(64)
HSPIPELINE	Integer
HSPIPELINESTAGE	Integer
ISDELETED	Boolean
OBJECTID	Integer
PORTALID	Integer
SUBJECT	VarChar(64)

TICKETS_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/tickets
```

Columns

The TICKETS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(9)

Column Name	Data Type
CREATEDBY	VarChar(64)
HSPIPELINE	Integer
HSPIPELINESTAGE	Integer
PROPERTIES_CONTENT	VarChar(64)
PROPERTIES_SUBJECT	VarChar(64)
UPDATEDAT	Timestamp(9)

URLMAPPINGS

Endpoint

`https://api.hubapi.com/url-mappings/v3/url-mappings`

Columns

The URLMAPPINGS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CONTENTGROUPID	VarChar(64)
CREATEDAT	Timestamp(3)
DELETEDAT	Timestamp(3)
DESTINATION	VarChar(64)
ISMATCHFULLURL	Boolean
ISMATCHQUERYSTRING	Boolean
ISONLYAFTERNOTFOUND	Boolean
NAME	VarChar(64)
PORTALID	Integer
PRECEDENCE	Integer
REDIRECTSTYLE	Integer

Column Name	Data Type
ROUTEPREFIX	VarChar(256)
UPDATEDAT	Timestamp(3)

WORKFLOW

Endpoint

```
https://api.hubapi.com/automation/v3/workflows/{WorkflowId}
```

Columns

The WORKFLOW table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
WORKFLOWID*	BigInt
ALLOWCONTACTTOTRIGGERMULTIPLETIMES	Boolean
ID	Integer
ISENABLED	Boolean
ISINTERNAL	Boolean
ISLISTENING	Boolean
NAME	VarChar(64)
ONLYEXECONBIZDAYS	Boolean
UPDATEDAT	Timestamp(3)