Progress DataDirect for ODBC for SAP S/4HANA

User's Guide

Release 8.0.1

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for SAP S/4HANA Driver

The Progress® DataDirect® for ODBC™ for SAP S/4HANA™ driver supports SQL read-write access for ODBC applications to SAP ODATA V2 services exposed through the SAP Gateway, including S/4HANA and BW/4HANA. To support SQL access, the driver creates a relational map of the SAP data model and translates SQL statements to OData requests. In addition, the driver employs a SQL engine component that provides support to SQL constructs unavailable in SAP services. This functionality offers a number of advantages, including support for reporting data and metadata in a form that ODBC applications are ready to use.

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:

- Driver specifications
- Additional information
- Troubleshooting
- Contacting Technical Support

What's new in this release?

Support and Certifications

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

Changes for 8.0.1 GA

- **Enhancements**
  - The driver is now compiled with a Visual Studio 2022 compiler for the Windows platforms. As a result, you must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher on your machine to run the driver.
  - The driver has been enhanced to support SAP BW/4HANA and other SAP ODATA V2 services exposed through the SAP Gateway.
    - For details, see Supported SQL statements and extensions on page 89.
    - For information on SAP services that use ODATA V2 APIs, refer to the SAP API documentation: ODATA V2.
  - The driver has been enhanced to support CDS views that require parameters. For details, see CDS views on page 21.
  - The driver has been enhanced to support HTTP Header authentication for use with the S/4HANA sandbox hosted at `https://api.sap.com`. See HTTP header authentication (S/4HANA only) on page 47 for details.
  - The driver has been enhanced to support a default value of 100000 and a maximum value of `1000000` for the connection option Web Service Fetch Size on page 87.
  - The driver has been enhanced to determine the maximum size of VARCHAR columns within the result set metadata using the new Maximum Varchar Size on page 72 connection option.
  - A Password Encryption Tool, called ddencpwd, is now included with the product package. It encrypts passwords for secure handling in connection strings and odbc.ini files. At connection, the driver decrypts these passwords and passes them to the data source as required. See Password Encryption Tool (UNIX/Linux only) on page 44 for details.

Highlights of 8.0.0 Release

- The driver supports SQL read-write access to SAP S/4HANA. For details, see Supported SQL statements and extensions on page 89 and Introduction to the SAP S/4HANA data model on page 115.

- 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 S/4HANA data types through data type inference. See Data types on page 20 for details.
- The driver supports custom fields, including many fields added through third-party plug-ins.
- The driver supports Basic authentication. See Basic authentication on page 46 for details.
- The driver supports the handling of large result sets with paging, and the Fetch Size (FetchSize) and Web Service Fetch Size (WSFetchSize) connection options. See Fetch Size on page 66 and Web Service Fetch Size on page 87 for details.
- On Windows platforms, the driver includes a redesigned ODBC setup dialog, the DataDirect ODBC Driver Configuration Manager, for quick configuration and testing of your driver. The Configuration Manager allows you to:
  - Configure data sources
  - Generate and edit connection URLs
  - Test connect data sources and connection URLs
  - Execute SQL commands for testing
  - Access connection option descriptions and the full product documentation

  See Configuring data sources with the Configuration Manager on page 35 and Generating connection strings with the Configuration Manager on page 37 for details.

Driver requirements

Data source and platform requirements

For the latest support information, visit the DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf.

Java requirements

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

Windows requirements for 32-bit drivers

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

Windows requirements for 64-bit drivers

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

Linux requirements for 32-bit drivers

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

Linux requirements for 64-bit drivers

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

Installing and setting up the driver (Windows)

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

To begin accessing data with the driver:

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

      `PROGRESS_DATADIRECT_ODBC_`*`nn`*`_WIN_`*`xx`*`_INSTALL.exe`

   c) Follow the prompts to complete installation.

   Note:

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

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

3. To configure the driver using the ODBC Administrator (GUI), start the ODBC Administrator from the Progress DataDirect program group. The GUI dialog allows you to configure the data source definitions in the Windows Registry or generate connection strings.

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

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

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

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

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

   - **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`.
   - **Host Name**: Type the name or the IP address of the server to which you want to connect. For example,

     S/4HANA:

     `https://mycompany.s4hana.ondemand.com`.

     BW/4HANA:

     `https://myinstance.company.com`
   - **Service List**: (Optional) Type a comma-separated list of standard and/or custom services to which the driver connects. For example, `API_BUSINESS_PARTNER,API_SALES_ORDER_SRV`. If no value is specified for this option, the driver attempts to discover all the services included in your communication scenario or to which you have access, which can adversely affect performance at connection.
   - **User**: Type the user name that is used to connect to the service.
   - **Password**: Type the password that is used to connect to the service.

   Note: User and Password connection options can also be specified in the logon dialog box or passed by your application.

   Note: See Basic authentication on page 46 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 16 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 options by functionality.
- Configuring data sources with the Configuration Manager on page 35 guides you through using the GUI to configure the driver.
- Performance considerations on page 48 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. The logon dialog appears. If not already specified, update the fields provided; then, click **OK**.

   The information you provide depends on the authentication method you are using to connect. The following authentication methods are supported:

   - Basic authentication on page 46
   - HTTP header authentication (S/4HANA only) on page 47 (only for S/4HANA sandbox hosted at `https://api.sap.com`)

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

9. If the test was successful, the window displays a confirmation message.
10. Click **OK** to close the setup dialog. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the Configuration Manager to modify your data source, or you can override these defaults by connecting to the data source using a connection string with alternate values.
11. Connect to your instance 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 89: 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.

Installing and setting up the driver (Linux)

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

To begin accessing data with the driver:

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

      `PROGRESS_DATADIRECT_ODBC_`*`nn`*`_LINUX_`*`xx`*`_INSTALL.bin`

   c) Follow the prompts to complete installation.

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

2. Configure the environment variables:
   a) Check your permissions. You must log in as a user with full r/w/x permissions recursively on the entire product installation directory.
   b) Run one of the following product setup scripts from the installation directory to set variables: `odbc.sh` or `odbc.csh`. For Korn, Bourne, and equivalent shells, execute `odbc.sh`. For a C shell, execute `odbc.csh`. Executing the setup script:

      - Sets the ODBCINI environment variable to point to the path from the root directory to the system information file where your data source resides. For details, see "ODBCINI."
      - Sets the library path environment variable for your Linux operating system, `LD_LIBRARY_PATH`, to include the directory containing your JVM's `libjvm.so` file. For details, see "Library search path."

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

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

     ```
     [ODBC Data Sources]
     SAP S4HANA=DataDirect 8.0 SAP S4HANA


     [SAP S4HANA]
     Driver=<install_dir>/lib/xxs4hana28.yy
     ...
     Description=My S/4HANA Data Source
     ...
     HostName=https://mycompany.s4hana.ondemand.com
     //For BW/4HANA: HostName= https://myinstance.company.com
     ...
     User=jsmith
     ...
     Password=secret
     ...
     ```

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

Note: The User and Password options are not required to be stored in the data source. They can also be sent separately by the application using the SQLConnect ODBC API. For SQLDriverConnect and SQLBrowseConnect, they will need to be specified in the data source or connection string.

- **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, DSN-less connections, for more information. For examples, see Connection string examples.

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

4. Set the values for any additional options that you want to configure. For additional information on optional features and functionality, see the following resources:
   - Connection string examples provides connection string examples that can be used to configure common functionality and features. You can modify and combine these examples to create a string that best suites your environment.

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

   - Connection option descriptions provides a complete list of supported options by functionality.
   - Performance Considerations describes connection options that affect performance, along with recommended settings.
5. Connect to your instance 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 89: 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.

Connection string examples

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

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

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

See also

Using a connection string on page 37

Basic authentication

This string includes the options used to connect using basic user name and password authentication.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see Using a connection string on page 37.

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=host_name;ServiceList=service[,service[,...]];
User=user_name;Password=password;[attribute=value[;...]];
```

where:

host_name

specifies the base URL with either the IP address or the domain name of the service to which you want to issue requests. For example, for S/4HANA, it can be either `https://mycompany.s4hana.ondemand.com` or `http://123.456.7.8`.

user_name

specifies the user name that is used to connect to the service. For example, `JSMITH`.

service

(optional) specifies the standard and/or custom service(s) to which the driver connects.

Note: Service List (ServiceList) allows you to limit the services to which the driver connects, thereby eliminating the discovery process and improving performance at connection. In addition, this option provides a method to connect to services not included in your communication scenario.

password

specifies the password used to connect to your service.

attribute=value

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

Note: The User and Password options are not required to be stored in the connection string. They can also be sent separately by the application using the SQLConnect ODBC API. For SQLDriverConnect and SQLBrowseConnect, they will need to be specified in the connection string.

Note: The following example connection string includes the options required for connecting with basic authentication:

For S/4HANA:

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://mycompany.s4hana.ondemand.com;
ServiceList=API_BUSINESS_PARTNER,API_SALES_ORDER_SRV;User=JSMITH;Password=secret;
```

For BW/4HANA:

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://myinstance.company.com;
ServiceList=API_BUSINESS_PARTNER,API_SALES_ORDER_SRV;User=JSMITH;Password=secret;
```

See also

Basic authentication on page 46
Connection option descriptions on page 57

HTTP header authentication (S/4HANA only)

This string includes the options used to connect using HTTP header authentication. HTTP header authentication can only be used to authenticate with the S/4HANA sandbox hosted at `https://api.sap.com`.

Note: The strings shown in this section use the DSN-less format. For additional formats, see Using a connection string on page 37.

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=host_name;AuthenticationMethod=25;
AuthHeader=api_key;SecurityToken=security_token;ServiceList=service[,service[,...]];
[attribute=value[;...]];
```

where:

host_name

specifies the URL of the S/4HANA sandbox instance: `https://api.sap.com`.

api_key

specifies the name of the HTTP header used to authenticate to the SAP S/4HANA instance.

security_token

specifies the token that is used to authenticate to the SAP S/4HANA instance.

service

(optional) specifies the standard and/or custom service(s) to which the driver connects.

Note: Service List (ServiceList) allows you to limit the services to which the driver connects, thereby eliminating the discovery process and improving performance at connection. In addition, this option provides a method to connect to services not included in your communication scenario.

---

attribute=value

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

Note: The security token option is not required to be stored in the connection string. It can also be sent separately by the application using the SQLConnect ODBC API. For SQLDriverConnect and SQLBrowseConnect, it will need to be specified in the connection string.

The following example connection string includes the options required for connecting with HTTP header authentication:

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://api.sap.com;
AuthenticationMethod=25-HttpHeader;AuthHeader=X-Api-Key;SecurityToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk31;
ServiceList=API_BUSINESS_PARTNER,API_SALES_ORDER_SRV;
```

See also

HTTP header authentication (S/4HANA only) on page 47
Connection option descriptions on page 57

Proxy server

This string includes the options you may need to connect through a proxy server with basic authentication.

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

This string includes the options used to connect through a proxy server with authentication.

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=host_name;ProxyHost=proxy_host;
ProxyPassword=proxy_password;ProxyPort=proxy_port;ProxyUser=proxy_user;
ServiceList=service[,service[,...]];User=user_name;Password=password;[attribute=value[;...]];
```

where:

host_name

specifies the base URL with either the IP address or the domain name of the service to which you want to issue requests. For example, for S/4HANA, it can be either `https://mycompany.s4hana.ondemand.com` or `http://123.456.7.8`.

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.

service

(optional) specifies the standard and/or custom service(s) to which the driver connects.

Note: Service List (ServiceList) allows you to limit the services to which the driver connects, thereby eliminating the discovery process and improving performance at connection. In addition, this option provides a method to connect to services not included in your communication scenario.

user_name

specifies the user name that is used to connect to the service. For example, `JSMITH`.

password

specifies the password used to connect to the service.

attribute=value

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

Note: The User and Password options are not required to be stored in the connection string. They can also be sent separately by the application using the SQLConnect ODBC API. For SQLDriverConnect and SQLBrowseConnect, they will need to be specified in the connection string.

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

For S/4HANA:

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://mycompany.s4hana.ondemand.com;
ProxyHost=pserver;ProxyPassword=proxysecret;ProxyPort=808;ProxyUser=admin1;
ServiceList=API_BUSINESS_PARTNER,API_SALES_ORDER_SRV;User=JSMITH;Password=secret;
```

For BW/4HANA:

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://myinstance.company.com;
ProxyHost=pserver;ProxyPassword=proxysecret;ProxyPort=808;ProxyUser=admin1;
ServiceList=API_BUSINESS_PARTNER,API_SALES_ORDER_SRV;User=JSMITH;Password=secret;
```

See also

Using a connection string on page 37
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.

Note: This data type mapping is supported for all SAP ODATA V2 services exposed through the SAP Gateway, including S/4HANA and BW/4HANA.

Table 1: Data Types

Data Type	ODBC Data Type
BINARY	SQL_VARBINARY
BOOLEAN	SQL_BIT
BYTE	SQL_SMALLINT
DATETIME	SQL_TYPE_TIMESTAMP
DECIMAL	SQL_DECIMAL
DOUBLE	SQL_DOUBLE
GUID	SQL_CHAR
INT16	SQL_SMALLINT
INT32	SQL_INTEGER
INT64	SQL_BIGINT
SBYTE	SQL_TINYINT
SINGLE	SQL_REAL
STRING	SQL_VARCHAR
TIMEOFDAY	SQL_TYPE_TIME

CDS views

The driver supports CDS views and CDS views that require parameters. The driver exposes CDS views as tables in a schema, as it does with other SAP services that use the ODATA V2 API. How a CDS view is queried depends on whether the CDS view requires parameters.

CDS views that do not require parameters

If querying a CDS view that does not require parameters, two-part notation should be used to specify the CDS view. In the following example, `YY1_CDSVIEW_SCHEMA` is the name of the schema and `CDSVIEW_ENTITY` is the entity or table name of the CDS view.

```
SELECT * FROM YY1_CDSVIEW_SCHEMA.CDSVIEW_ENTITY
```

In the following SQL statement, a `WHERE` clause is used to fetch records for the given region.

```
SELECT * FROM YY1_CDSVIEW_SCHEMA.CDSVIEW_ENTITY WHERE REGION = 'Northwest'
```

CDS views that require parameters

For a CDS view that requires a parameter, the required parameter must be specified in a WHERE clause and the values of the parameter must be specified in an equality expression. For example:

```
SELECT * FROM YY1_CDSVIEW_SCHEMA.CDSVIEW_ENTITY WHERE P_KEYDATE = {d '2017-03-28'}
```

You can determine which tables require WHERE clause filtering by querying the system table SYSTEM_COLUMNS_REQUIRED_FILTERS that resides in the driver's internal schema INFORMATION_SCHEMA. The following example searches the schema YY1_CDSVIEW_SCHEMA for this information.

```
SELECT * FROM INFORMATION_SCHEMA.SYSTEM_COLUMNS_REQUIRED_FILTERS WHERE TABLE_SCHEM =
'YY1_CDSVIEW_SCHEMA'
```

Note:

- Some CDS views may require more than one parameter. In these instances, each required parameter must be included in the WHERE clause using an AND operator to separate the expressions.
- Some CDS views may have optional parameters. When an optional parameter is included in the WHERE clause, the driver includes the optional parameter in the request.
- Currently, the OR operator is not supported for separating parameter values.

See also

Supported SQL statements and extensions on page 89
Introduction to the SAP S/4HANA data model on page 115

Driver specifications

This section describes the general functionality supported by the driver.

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

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

  The driver supports only the following Level 2 functions:

  - SQLColumnPrivileges
  - SQLDescribeParam
  - SQLForeignKeys
  - SQLPrimaryKeys
  - SQLProcedures
  - SQLTablePrivileges

- **Unicode support**: The driver is fully Unicode enabled. On Linux platforms, the driver supports both UTF-8 and UTF-16. On Windows platforms, the driver supports UCS-2/UTF-16 only.

  Refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for details.

- **Isolation and lock levels**: Because transactions are not supported, the driver supports only the isolation level 0 (read uncommitted).

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

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

Additional information

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

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

Troubleshooting

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

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

Contacting Technical Support

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

https://www.progress.com/support

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

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

- Your number or the serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.

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

December 2021, 8.0.1 Release of Progress DataDirect for ODBC for SAP S/4HANA, 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 for:

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

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

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

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

Note:

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

To use the Example application:

1. After you have configured the data source, navigate to the `instal_dir\samples\example` directory.
2. Open the application using one of the following methods:
   - Running the application executable or binary:
     - On Windows, double-click the `Example.exe` file.
     - On Linux, run the `example` application.
   - Executing a command-line argument. For example:
     - `example` *`connection_string`*
     - `example "DSN" "UID" "PWD"`
     - `example` *`connection_string`* `"`*`sql_command_1`*`" ["`*`sql_command_2`*`" ...]`

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

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

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

Power BI (Windows only)

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

1. Navigate to the `\tools\Power BI` subdirectory of the Progress DataDirect installation directory; then, locate the installation batch file `install.bat`.
2. Run the `install.bat` file. The following operations are executed by running the `install.bat` file:
   - The Power BI connector file, `DataDirectS4HANA.pqx`, is copied to the following directory.

     Note: The `DataDirectS4HANA.pqx` file must be used for all SAP ODATA V2 services exposed through the SAP Gateway, including S/4HANA and BW/4HANA.

     ```
     %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 SAP S/4HANA Connector**.
5. Click **Connect**. Then, from the **Progress DataDirect SAP S/4HANA Connector** window, provide the following information. Then, click **OK**.
   - **Data Source:** Enter a name for the data source. For example, `SAP S/4HANA 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 SAP S4HANA.tdc`

   Note: The `S4HANA.tdc` file is used for all SAP ODATA V2 services exposed through the SAP Gateway, including S/4HANA and BW/4HANA.

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.

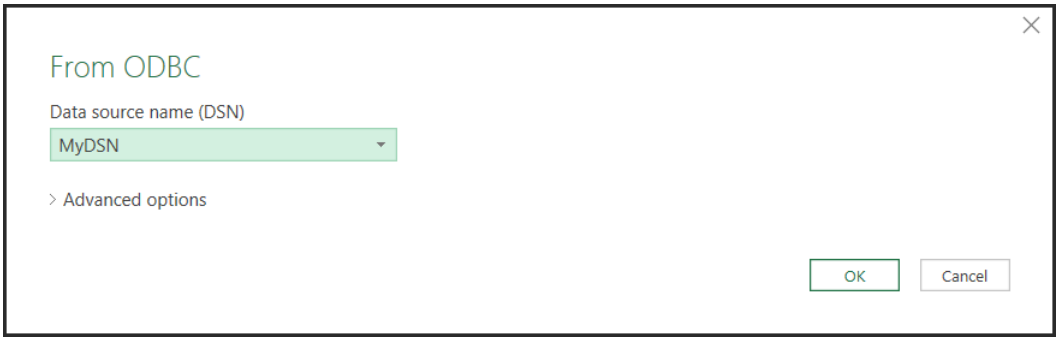
Microsoft Excel (Windows only)

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.

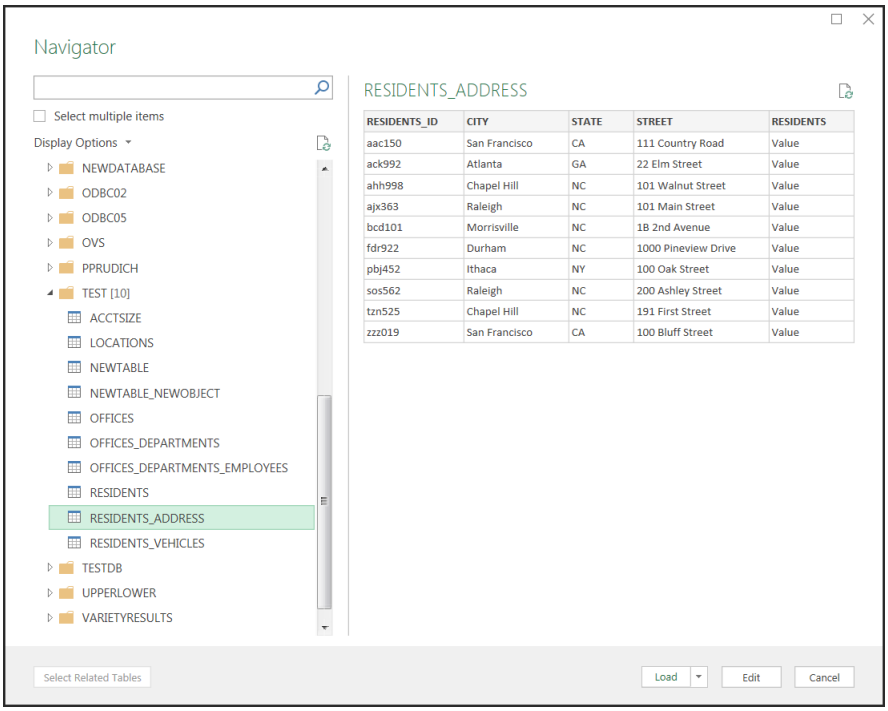


Select your data source from the Data Source Name (DSN) drop down; then, click **OK**.

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.

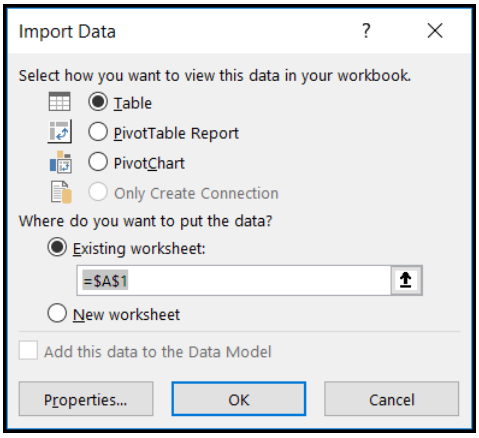


From the list, select the tables you want to access. A preview of your data will appear in the pane on the right. Optionally, click **Edit** to modify the results using the Query Editor. Refer to the Microsoft Excel product documentation for detailed information on using the Query Editor.

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:

- Authentication
- Performance considerations

Environment settings

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

Windows environment variables

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

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

Linux environment variables

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

Library search path

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

```
echo $SHELL
```

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

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

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

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

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

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

```
env
```

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

ODBCINI

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

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

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

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

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

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

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

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

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

See also

Configuration through the system information (odbc.ini) file on page 38

ODBCINST

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

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

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

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

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

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

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

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

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

See also

DSN-less connections on page 41

DD_INSTALLDIR

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

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

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

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

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

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

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

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

The next step is to test load the driver.

See also

Configuration through the system information (odbc.ini) file on page 38

The test loading tool

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

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

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

```
ivtestlib /opt/odbc/lib/ivs4hanaxx.so
```

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

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

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

UTF-16 applications on Linux

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

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

- The definition of SQLWCHAR in the ODBC header files must be switched from "char *" to "short *". To do this, the application uses #define SQLWCHARSHORT.
- The application must set the encoding for the environment or connection using one of the following attributes. If your application passes UTF-8 encoded strings to some connections and UTF-16 encoded strings to other connections in the same environment, encoding should be set for the connection only; otherwise, either method can be used.
  - To configure the encoding for the environment, set the ODBC environment attribute SQL_ATTR_APP_UNICODE_TYPE to a value of `SQL_DD_CP_UTF16`, for example:

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

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

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

Configuring data sources with the Configuration Manager

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

The driver includes an enhanced setup dialog, the Progress DataDirect SAP S/4HANA Configuration Manager, that allows you to configure data sources, generate connection strings, test connections, and execute test queries. On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using the Configuration Manager, as described in this section.

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

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

To configure and test a data source:

1. Open the Windows ODBC Administrator.
2. Open the Configuration Manager through the **User DSN** or **System DSN** tab.

   - **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.

     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the Configuration Manager.
   - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.

     If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the Configuration Manager.

   - **Note:** Configuring an existing or new file data source using the File DSN tab is not currently supported with the configuration manager.

   The SAP S/4HANA 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 instance with your settings. In the Test Connection window:

   a) Provide values for any fields required by your instance. 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**.

   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 38
Generating connection strings with the Configuration Manager on page 37
Connection option descriptions on page 57
Connection string examples on page 16

Generating connection strings with the Configuration Manager

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

The Progress DataDirect SAP S/4HANA Configuration Manager supports generating connection strings that can be used with your application. To generate a connection string, create a data source as described in "Configuring data sources with the Configuration Manager." As you provide connection option values, the Configuration Manager generates a connection string in the **Connection String** pane that corresponds to the data source.

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

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

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

See also

Configuring data sources with the Configuration Manager on page 35
Testing connections and queries with the Configuration Manager on page 44

Using a connection string

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

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

The DSN-less connection string has the form:

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

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

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

```
DSN=SAP S4HANA;USER=JOHN;PWD=XYZZY
```

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

```
FILEDSN=S4HANA;USER=JSMITH;PWD=XYZZY
```

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

S/4HANA:

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://mycompany.s4hana.ondemand.com;
User=JSMITH;Password=secret;
```

BW/4HANA:

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://myinstance.company.com;
User=JSMITH;Password=secret;
```

See also

Connection option descriptions on page 57
Connection string examples on page 16

Additional configuration methods for Linux

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

Configuration through the system information (odbc.ini) file

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

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

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

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

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

```
S4HANA=DataDirect 8.0 SAP S4HANA
```

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, `[S4HANA]`. 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*.

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]
S4HANA=DataDirect 8.0 SAP S4HANA

[S4HANA]
Driver=ODBCHOME/lib/ivs4hana28.so
Description=DataDirect 8.0 SAP S4HANA Driver
ApplicationUsingThreads=1
AuthenticationMethod=0
AuthHeader=
CreateMap=0
ExtendedOptions=
FetchSize=100
HostName=
InitializationString=
JVMArgs=-Xmx256m
JVMClasspath=
JVMPath=
KeywordConflictSuffix=
LogConfigFile=ddlogging.properties
MaxVarcharSize=64000
Password=
ProxyHost=
ProxyPassword=
ProxyPort=0
ProxyUser=
ReadOnly=0
RefreshSchema=0
ReportCodepageConversionErrors=0
SchemaMap=
SecurityToken=
ServerPortNumber=19954
ServerProxyHost=
ServerProxyPort=
ServerProxyUser=
ServiceList=
SQLEngineMode=0
StmtCallLimit=0
StmtCallLimitBehavior=1
TransactionMode=0
User=
WSCompressData=0
WSFetchSize=100000

[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.

  Note: In addition to a text editor, you can also modify the `odbc.ini` file using the Configuration Manager. See "Configuring data sources with the Configuration Manager" for details.

  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 SAP S4HANA=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 33
Configuration through the system information (odbc.ini) file on page 38

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 SAP S4HANA=Installed

[DataDirect 8.0 SAP S4HANA]
Driver=ODBCHOME/lib/ivs4hana28.so
JarFile=ODBCHOME/java/lib/s4hana.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
```

Note: The `s4hana.jar` and `ivs4hana28.so` are used for all SAP ODATA V2 services exposed through the SAP Gateway, including S/4HANA and BW/4HANA.

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:

S/4HANA

```
[ODBC]
Driver=DataDirect 8.0 SAP S4HANA
HostName=https://mycompany.s4hana.ondemand.com
User=JSMITH
Password=secret
```

BW/4HANA

```
[ODBC]
Driver=DataDirect 8.0 SAP S4HANA
HostName=https://myinstance.company.com
User=JSMITH
Password=secret
```

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\S4HANA.dsn
```

or, on Linux:

```
FILEDSN=/home/users/john/filedsn/S4HANA.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/S4HANA.dsn;User=john;PWD=test01
```

See also

Configuring and connecting to data sources on page 31
DSN-less connections on page 41
Configuration through the system information (odbc.ini) file on page 38

Testing connections and queries with the Configuration Manager

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

You can quickly test data sources, connection strings and queries using Progress DataDirect SAP S/4HANA Configuration Manager.

To test your connection and query:

1. Open the Windows ODBC Administrator. Then, select or add a data source to open the SAP S/4HANA Configuration Manager.

   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 SAP S/4HANA 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 instance 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 instance.
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**.

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

Password Encryption Tool (UNIX/Linux only)

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

- KeyPassword
- KeyStorePassword
- TrustStorePassword

- Password

To use the Password Encryption Tool:

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

   `ddencpwd` *`password`*

   where:

   `password`

   is the password you want to encrypt.

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

   `KeyPassword=`*`returned_value`*

4. Repeat Steps 2 and 3 to encrypt additional passwords.
5. If using an `odbc.ini` file, save your file.

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

Using a logon dialog box

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

In the dialog box, provide the following information:

- In the Host Name field, specify the base URL of the service to which you want to issue requests.
- In the User field, type the user name that is used to connect to the service.
- In the Password, type the password used to connect to the service.

Authentication

The driver supports the following authentication methods:

- *Basic authentication* authenticates using the specified user IDs and passwords.
- (S/4HANA only) *HTTP Header authentication* authenticates by passing security tokens via the HTTP headers. This type of authentication can only be used to authenticate with the S/4HANA sandbox hosted at `https://api.sap.com`.

By default, the driver is configured to use basic authentication (`AuthenticationMethod=0`).

See also

Authentication Method on page 63

Basic authentication

The driver supports basic user ID and password authentication.

To configure the driver to use basic authentication:

- Set the Host Name (HostName) option to the IP address or domain name of the service to which you want to connect. For example, for S/4HANA, it can be either `https://mycompany.s4hana.ondemand.com` or `http://123.456.7.8`.
- Set the User (User) option to specify the name of the user connecting to the service.
- Set the Password (Password) option to specify the password for the user connecting to the service.

Note: The User and Password options are not required to be stored in the connection string. They can also be sent separately by the application using the SQLConnect ODBC API. For SQLDriverConnect and SQLBrowseConnect, they will need to be specified in the connection string.

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

Connection string for S/4HANA

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://mycompany.s4hana.ondemand.com;
User=jsmith;Password=secret;
```

Connection string for BW/4HANA

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://myinstance.company.com;
User=JSMITH;Password=secret;
```

odbc.ini for S/4HANA

```
[SAP S4HANA]
Driver=ODBCHOME/lib/ivs4hana28.so
...
Description=DataDirect 8.0 SAP S4HANA
...
HostName=https://mycompany.s4hana.ondemand.com
...
User=JSMITH
...
Password=secret
...
```

odbc.ini for BW/4HANA

```
[SAP BW4HANA]
Driver=ODBCHOME/lib/ivs4hana28.so
...
Description=DataDirect 8.0 SAP BW4HANA
...
HostName=https://myinstance.company.com
...
User=JSMITH
...
```

```
Password=secret
...
```

See also

Connection option descriptions on page 57

HTTP header authentication (S/4HANA only)

HTTP header authentication can be used to authenticate with the S/4HANA sandbox hosted at `https://api.sap.com`.

To configure the driver to use HTTP header authentication:

- Set the Host Name (HostName) option to the S/4HANA sandbox instance: `https://api.sap.com`.
- Set the Authentication Method (AuthenticationMethod) option to `25` (HTTP Header).
- Set the AuthHeader (`AuthHeader`) option to specify the name of the HTTP header used for authentication. The default is `Authorization`.
- Set the Security Token option to specify the security token for the user connecting to the service.
- Optionally, specify values for any additional options you want to configure. See "Connection option descriptions" for a complete list of options.

Note: The Security Token option is not required to be stored in the connection string. It can also be passed separately by the application.

The following example show the connection information required to establish a session with HTTP header authentication enabled and AuthHeader is set to the default.

Connection string

```
DRIVER=DataDirect 8.0 SAP S4HANA;HostName=https://api.sap.com;
AuthenticationMethod=25-HttpHeader;AuthHeader=X-Api-Key;
SecurityToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

odbc.ini

```
[SAP S4HANA]
Driver=ODBCHOME/lib/ivs4hana28.so
...
Description=DataDirect 8.0 SAP S4HANA
...
HostName=https://api.sap.com
...
AuthenticationMethod=25
...
AuthHeader=X-Api-Key
...
SecurityToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831
...
```

See also

HTTP header authentication (S/4HANA only) on page 18
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.

CreateMap: CreateMap determines whether the driver creates the internal files required for a relational map of the native data when establishing a connection. When the driver creates these internal files, an initial connection can take a few minutes, depending on network speeds and the amount of metadata the driver must retrieve from the service. For example, when CreateMap is set to OnChange (the default) and changes have been made to the back-end schema, the connection may be delayed because the driver runs a discovery on the schema and overwrites the internal files used to create a relational map of the data.

Fetch Size/Web Service Fetch Size (FetchSize/WSFetchSize): The connection options Fetch Size and Web Service Fetch Size can be used to adjust the trade-off between throughput and response time. In general, setting larger values for Web Service Fetch Size and Fetch Size will improve throughput, but can reduce response time.

For example, if an application attempts to fetch 100,000 rows from the remote data source and Web Service Fetch Size is set to 500, the driver must make 200 Web service calls to get the 100,000 rows. If, however, Web Service Fetch Size is set to 4000, the driver only needs to make 25 Web service calls to retrieve 100,000 rows. Web service calls are expensive, so generally, minimizing Web service calls increases throughput. In addition, many Cloud data sources impose limits on the number of Web service calls that can be made in a given period of time. Minimizing the number of Web service calls used to fetch data also can help prevent exceeding the data source call limits.

For many applications, throughput is the primary performance measure, but for interactive applications, such as Web applications, response time (how fast the first set of data is returned) is more important than throughput. For example, suppose that you have a Web application that displays data 50 rows to a page and that, on average, you view three or four pages. Response time can be improved by setting Fetch Size to 50 (the number of rows displayed on a page) and WSFetch Size to 200. With these settings, the driver fetches all of the rows from the remote data source that you would typically view in a single Web service call and only processes the rows needed to display the first page.

Service List (ServiceList): If your application does not require connecting to all the services to which you have access, you can improve performance using the Service List option. By default, the driver attempts to discover and connect to all the services to which it has access. However, you can limit the number of services the driver attempts to connect to by specifying only the required services with this option. This eliminates the discovery process and reduces the overhead associated with connecting to services you do not use, thereby improving performance.

Service Version (ServiceVersion): When there are multiple versions of a service, Service Version determines which versions the driver returns metadata for. If you only require metadata for the latest version of a service, set this option to `0` (Latest), the default. This improves performance at connection by eliminating the overhead associated with fetching metadata for multiple versions.

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

See also

Connection option descriptions on page 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. Before starting the SQL engine server, choose a directory to store the local database files using the Schema Map (SchemaMap) option. Make sure that you have the correct permissions to write to this directory.

Note: The Configuration Manager is currently supported only on Windows platforms. To configure the SQL engine on Linux platforms, see "Configuring the SQL engine server using Java options."

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: `19954`

     64-bit: `19953`
   - **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\s4hana.jar}`

     Linux: `{.:/home/user1/`*`install_dir`*`/java/lib/s4hana.jar}`
   - **JVM Arguments**: A string that contains the arguments that are passed to the JVM that the driver is starting. The location of the JVM must be specified on your PATH. See "JVM Arguments" for details. The default value is:

     `-Xmx1024m`
   - **JVM Path**: Specifies fully qualified path to the JVM executable that you want to use to run the SQL engine server. The path must not contain double quotation marks.
4. Optionally, if connecting through a proxy server, provide values for the following options:
   - **Server Proxy Host**: Specifies the host name of the proxy server used by the SQL engine. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.

- **Server Proxy Port**: Specifies the port needed to connect to the proxy server used by the SQL engine.
- **Server Proxy User**: Specifies the user name needed to connect to the proxy server used by the SQL engine.
- **Server Proxy Password**: Specifies the password needed to connect to the proxy server used by the SQL engine.

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

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

See also

JVM Arguments on page 68

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 using the driver. Before starting the SQL engine server, verify that you have the correct permissions to write to the directory specified by the Schema Map (SchemaMap) option.

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

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

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

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

For example:

```
java -Xmx1024m -cp "/opt/Progress/DataDirect/ODBC_80_64bit/java/lib/s4hana.jar"
        com.ddtek.phoenix.sql.Server -port 19953 -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/s4hana.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: • `19954` (32-bit drivers) • `19953` (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.

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

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

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

General options

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 (n/a)	No default value
Hostname (host)	No default value
ServiceList (slist)	No default value

Basic authentication options

The following table summarizes options used for basic user id and password authentication.

Table 4: Basic authentication options

Attribute (Short Name)	Default
AuthenticationMethod (am)	0 (Basic)
Hostname (host)	No default value
Password (pwd)	No default value
User	No default value

HTTP Header authentication options (S/4HANA only)

The following table summarizes the connection options required to connect to a service when using HTTP Header authentication. This authentication method is supported only for S/4HANA services.

Table 5: HTTP Header authentication options

Attribute (Short Name)	Default
AuthenticationMethod (am)	25 (HTTPHeader)
AuthHeader (ah)	No default value
Hostname (host)	No default value
SecurityToken (st)	No default value

Mapping options

The following table summarizes connection options involved in mapping the SAP S/4HANA data model to a SQL model.

Table 6: Mapping options

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

SQL engine options

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

Table 7: 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\s4hana.jar`
JVMPath (jvmp)	*`install_dir`*`\jre\bin\java.exe`
ServerPortNumber (spn)	32-bit: `19954` 64-bit: `19953`
ServerProxyHost (sph)	No default value
ServerProxyPassword (spw)	No default value
ServerProxyPort (spp)	No default value
ServerProxyUser (spu)	No default value
SQLEngineMode (sem)	For Windows: `0` (Auto) For Linux: The SQL engine runs in Direct mode by default.
SQLService (ss)	No default value

Proxy server options

The following table summarizes proxy server connection options.

Table 8: Proxy server options

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

Web service options

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

Table 9: Web service options

Attribute (Short Name)	Default
StmtCallLimit (scl)	`0` (no limit)
StmtCallLimitBehavior (sclb)	`1` (ErrorAlways)
WSCompressData (wscd)	`0` (Compress)
WSFetchSize (wsfs)	`100000`

Additional options

The following table summarizes additional connection options.

Table 10: Additional options

Attribute (Short Name)	Default
ApplicationUsingThreads (aut)	`1` (True)
ExtendedOptions (xo)	No default value
FetchSize (fs)	`100` (rows)

Attribute (Short Name)	Default
InitializationString (is)	No default value
LogConfigFile (lgcf)	`ddlogging.properties`
MaxVarcharSize (mxvs)	`64000`
ReadOnly (ro)	`0` (false)
ReportCodepageConversionErrors (rcce)	`0` (Ignore Errors)
ServiceVersion (sver)	`0` (Latest)
TransactionMode (tm)	`0` (NoTransactions)

For details, see the following topics:

- Application Using Threads
- Authentication Header
- Authentication Method
- Create Map
- Data Source Name
- Description
- Extended Options
- Fetch Size
- Host Name
- Initialization String
- JVM Arguments
- JVM Classpath
- JVM Path
- Keyword Conflict Suffix
- Log Config File
- Maximum Varchar Size
- Password
- Proxy Host
- Proxy Password
- Proxy Port

- Proxy User
- Read Only
- Refresh Schema
- Report Codepage Conversion Errors
- Schema Map
- Security Token
- Server Port Number
- Server Proxy Host
- Server Proxy Password
- Server Proxy Port
- Server Proxy User
- Service List
- Service Version
- SQL Engine Mode
- SQL Service
- Statement Call Limit
- Statement Call Limit Behavior
- Transaction Mode
- User
- Web Service Compress Data
- Web Service Fetch Size

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 48

Authentication Header

Attribute

AuthHeader (ah)

Purpose

Specifies the name of the HTTP header used for authentication. This option is used when Header-based token authentication (`AuthenticationMethod=25`) is enabled; otherwise, this option is ignored.

Valid Values

`auth_header`

where:

`auth_header`

is the name of the HTTP header used for authentication. For example, `X-Api-Key`.

Default Value

No default value

Authentication Method

Attribute

AuthenticationMethod (am)

Purpose

Determines which authentication method the driver uses during the course of a session.

Valid Values

`0` | `25`

Behavior

If set to `0` (Basic), the driver uses a hashed value, based on the concatenation of the user name and password, for authentication.

(S/4HANA only) If set to `25` (HTTPHeader), the driver passes security tokens via HTTP headers for authentication. You must also configure `Security Token` option and the `Authentication Header` option.

Default Value

`0` (Basic)

See also

Basic authentication on page 46

Create Map

Attribute

CreateMap (cm)

Purpose

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

Valid Values

`0` | `1` | `2` | `3`

Behavior

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

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

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

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

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

Notes

- The internal files share the same directory as the schema map configuration file. This directory is specified with the Schema Map connection option.
- You can refresh the internal files related to an existing relational view of your data with the SQL extension Refresh Map. Refresh Map runs a discovery against your native data and updates your internal files accordingly.

Default Value

3 (Session)

See also

Schema Map on page 77
Refresh Map (EXT) on page 91

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

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 48

Host Name

Attribute

HostName (host)

Purpose

Specifies the base URL of the service to which you want to connect. The base URL is comprised of either the domain name or the IP address of the service.

Valid Values

`string`

where:

`string`

is the base URL of the service to which you want to connect. For example:

- For SAP S/4HANA:
  - Base URL with a domain name: `https://mycompany.s4hana.ondemand.com`

- Base URL with an IP address: `http://123.456.7.8`

- For SAP BW/4HANA:
  - Base URL with a domain name: `https://myinstance.company.com`
  - Base URL with an IP address: `http://123.456.7.8`

Notes

- The ServerName attribute is an alias for the Host Name (HostName) option.

Default Value

No default value

Initialization String

Attribute

InitializationString (is)

Purpose

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

Valid Values

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

where:

command

is a SQL command.

Notes

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

Default Value

No default value

JVM Arguments

Attribute

JVMArgs (jvma)

Purpose

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

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

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

Valid Values

String

where:

String

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

Example

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

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

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

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

Default Value

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

```
-Xmx256m
```

For all other configurations:

```
-Xmx1024m
```

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 S/4HANA and MongoDB drivers on Windows, you would specify a value of `{c:\install_dir\java\lib\s4hana.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\s4hana.jar}
```

On Linux:

```
{.:/home/user1/install_dir/java/lib/s4hana.jar}
```

Default Value

install_dir`\java\lib\s4hana.jar`

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`

Keyword Conflict Suffix

Attribute

KeywordConflictSuffix (kcs)

Purpose

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

Valid Values

String

where:

String

is a string of up to 5 alphanumeric characters.

Example

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

Default Value

No default value

Log Config File

Attribute

LogConfigFile (lgcf)

Purpose

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

Valid Values

String

where:

String

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

Default Value

`ddlogging.properties`

Maximum Varchar Size

Attribute

MaxVarcharSize (mxvs)

Purpose

Determines the maximum size of VARCHAR columns when described within the result set metadata.

Valid Values

x

where:

x

is an integer greater than or equal to 1 and less than or equal to 2147483647.

Notes

When string functions are used within the Select list of a Select statement, the driver describes the resulting string value with a size of 2147483647. For instance, concatenating two columns that are each defined as VARCHAR(10) will result in a VARCHAR(2147483647) rather than a VARCHAR(20). The unnecessarily long size can result in undesirable behavior in some JDBC applications.

Default Value

`64000`

Password

Attribute

Password (pwd)

Purpose

A password that is used to connect to the instance.

Behavior

`password`

where:

`password`

is a valid password. The password is case-sensitive.

Default Value

No default value

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 19

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 19

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 server on page 19

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 19

Read Only

Attribute

ReadOnly (ro)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Default Value

1 (true)

Refresh Schema

Attribute

RefreshSchema (rs)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Notes

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

Default Value

`0` (false)

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 instance or to and from the application. The error or warning generated is `Code page conversion error encountered`. In the case of parameter data conversion errors, the driver adds the following sentence: `Error in parameter x`, where *x* is the parameter number. The standard rules for returning specific row and column errors for bulk operations apply.

Valid Values

`0` | `1` | `2`

Behavior

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

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

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

Default Value

`0`

Schema Map

Attribute

SchemaMap (smp)

Purpose

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

Valid Values

string

where:

string

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

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

Notes

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

Default Value

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

- Windows
  - DD_HOME environment variable
  - LOCALAPPDATA environment variable
  - APPDATA environment variable
  - User’s home directory
- For UNIX/Linux
  - DD_HOME environment variable

- User’s home directory

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

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

Security Token

Attribute

SecurityToken (st)

Purpose

Specifies the security token required to make a connection to the server. This option is required when token-based authentication is enabled (`AuthenticationMethod=25`); otherwise, this option is ignored. If an API key is required and you do not supply one, the driver returns an error indicating that an invalid user or password was supplied.

Valid Values

`string`

where:

`string`

is the value of the API key assigned to the user.

Notes

- The value of this option correlates to the 'API Key' value for the service.

Default Value

No default value

See also

HTTP header authentication (S/4HANA only) on page 18

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: 19954

64-bit: 19953

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

Service List

Attribute

ServiceList (slist)

Purpose

Specifies a comma-separated list of standard and/or custom services to which the driver connects. This option allows you to limit the services to which the driver connects, thereby improving performance at connection. In addition, this option provides a method to connect to services not included in your communication scenario.

Valid Values

service_name[,*service_name*,...]

where:

service_name

is the name of the service to which you want to connect.

Example

```
API_BUSINESS_PARTNER,API_SALES_ORDER_SRV
```

Notes

- If no value is specified for this option, the driver attempts to discover all the services included in your communication scenario or to which you have access.
- This option is recommended when you do not have access to the *SAP Analytics Cloud for Planning Integration (SAP_COM_0087)* communication scenario.

Default Value

No default value

Service Version

Attribute

ServiceVersion (sver)

Purpose

Determines which versions of a service the driver returns metadata for when the service has multiple versions.

Valid Values

`0` | `1`

Behavior

If set to `0` (Latest), the driver returns metadata only for the latest version of the service(s) to which it connects. For example, if the service has both V=001 and V=002, the driver returns metadata only for V=002.

If set to `1` (All), the driver returns metatdata for every version of the service to which it connects.

Notes

- If a service has only one version, the driver returns only the metadata for that version.
- If a value is specified for the Service List (ServiceList) option, the driver returns metadata only for the services and versions specified by the Service List and Service Version option.
- If metadata is required only for the latest version of a service, set this option to `0` (Latest), the default. This improves performance at connection by limiting the overhead associated with fetching metadata for multiple versions.

Default Value

`0` (Latest)

See also

Service List on page 81

SQL Engine Mode

Attribute

SQLEngineMode (sem)

Purpose

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

Valid Values

`0` | `1` | `2`

Behavior

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

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

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

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

Default Value

For Windows:

`1` (Server)

For Linux:

`2` (Direct)

See also

Using the SQL engine server on page 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

Statement Call Limit

Attribute

StmtCallLimit (scl)

Purpose

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

Valid Values

`0` | *x*

where:

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

Behavior

If set to `0`, there is no limit.

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

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

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

Default Value

`0`

See also

Statement Call Limit Behavior on page 85

Statement Call Limit Behavior

Attribute

StmtCallLimitBehavior (sclb)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Default Value

`1` (ErrorAlways)

See also

Statement Call Limit on page 84

Transaction Mode

Attribute

TransactionMode (tm)

Purpose

Specifies how the driver handles manual transactions.

Valid Values

`0` | `1` | `3`

Behavior

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

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

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

Default Value

3 (Full)

User

Attribute

User

Purpose

Specifies the user name that is used to connect to the instance.

Valid Values

String

where:

String

is a valid user name. The user name is case-insensitive.

Default Value

No default value

Web Service Compress Data

Attribute

WSCompressData (wscd)

Purpose

Specifies whether the driver compresses data it sends to or receives from the instance.

Valid Values

0 | 1

Behavior

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

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

Notes

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

Default Value

0 (Compress)

See also

Performance considerations on page 48

Web Service Fetch Size

Attribute

WSFetchSize (wsfs)

Purpose

Specifies the number of rows of data the driver attempts to fetch for each ODBC call.

Valid Values

x

where:

x

is a positive integer from 100 to 1000000 that defines a number of rows.

Behavior

If set to *x*, the driver attempts to fetch up to a maximum of the specified number of rows. Setting the value lower than the maximum can reduce the response time for returning the initial data; therefore, consider specifying smaller values for interactive applications.

Notes

Web Service Fetch Size and Fetch Size (FetchSize) 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.

Default Value

100000

See also

Performance considerations on page 48

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)
- Refresh Map (EXT)
- Select
- Update
- Subqueries
- SQL expressions

Alter Session (EXT)

Purpose

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

Syntax

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

where:

attribute_name

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

value

specifies the value for that attribute.

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

Table 11: 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=SAP S/4HANA`
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 s4hana.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()`

Refresh Map (EXT)

Purpose

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

Syntax

```
REFRESH MAP
```

Notes

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

Select

Purpose

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

Syntax

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

where:

select_clause

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

from_clause

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

where_clause

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

groupby_clause

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

having_clause

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

UNION

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

INTERSECT

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

EXCEPT | MINUS

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

orderby_clause

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

limit_clause

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

See also

Select clause on page 93
From clause on page 95
Where clause on page 96
Group By clause on page 97
Having clause on page 98
Union operator on page 98
Intersect operator on page 99
Except and Minus operators on page 100
Order By clause on page 101
Limit clause on page 101

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 105
Aggregate functions on page 94

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 12: 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.

See also

Subquery in a From clause on page 96

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.

See also

Subqueries on page 103

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

SQL expressions on page 105

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
```

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

Subqueries on page 103
SQL expressions on page 105

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

Subqueries on page 103
SQL expressions on page 105

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

Subqueries on page 103
SQL expressions on page 105

Limit clause

Purpose

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

Syntax

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

where:

number_of_rows

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

OFFSET

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

Notes

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

Example

The following example returns a maximum of 20 rows.

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

Update

Purpose

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

Syntax

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

table_name

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

column_name

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

expression

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

Example A

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

```
UPDATE emp SET salary=32000, exempt=1
WHERE emp_id = 'E10001'
```

Example B

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

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

Notes

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

See also

Where clause on page 96

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 13: 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`
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'`

SQL Type	Literal Syntax	Example
DECIMAL	*n.f* where: *n* is the integral part *f* is the fractional part	0.25 3.1415 -7.48
DOUBLE	*n.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 14: 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 15: 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 16: Comparison Operators

Operator	Purpose	Example
=	Equality test.	`SELECT * FROM emp WHERE sal = 1500`
!=<>	Inequality test.	`SELECT * FROM emp WHERE sal != 1500`

Operator	Purpose	Example
><	"Greater than" and "less than" tests.	`SELECT * FROM emp WHERE sal > 1500 SELECT * FROM emp WHERE sal < 1500`
>=<=	"Greater than or equal to" and "less than or equal to" tests.	`SELECT * FROM emp WHERE sal >= 1500 SELECT * FROM emp WHERE sal <= 1500`
LIKE	% and _ wildcards can be used to search for a pattern in a column. The percent sign denotes zero, one, or multiple characters, while the underscore denotes a single character. The right-hand side of a LIKE expression must evaluate to a string or binary.	`SELECT * FROM emp WHERE ENAME LIKE 'J%'`

ESCAPE clause in LIKE operator
LIKE 'pattern string' ESCAPE 'c'
	The Escape clause is supported in the LIKE predicate to indicate the escape character. Escape characters are used in the pattern string to indicate that any wildcard character that is after the escape character in the pattern string should be treated as a regular character.
	The default escape character is backslash (\).
		`SELECT * FROM emp WHERE ENAME LIKE 'J%\_%' ESCAPE '\'`
		This matches all records with names that start with letter 'J' and have the '_' character in them.
		`SELECT * FROM emp WHERE ENAME LIKE 'JOE\_JOHN' ESCAPE '\'`
		This matches only records with name 'JOE_JOHN'.

[NOT] IN	"Equal to any member of" test.	`SELECT * FROM emp WHERE job IN ('CLERK','ANALYST') SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30)`
[NOT] BETWEEN x AND y	"Greater than or equal to x" and "less than or equal to y."	`SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000`
EXISTS	Tests for existence of rows in a subquery.	`SELECT empno, ename, deptno FROM emp e WHERE EXISTS (SELECT deptno FROM dept WHERE e.deptno = dept.deptno)`
IS [NOT] NULL	Tests whether the value of the column or expression is NULL.	`SELECT * FROM emp WHERE ename IS NOT NULL SELECT * FROM emp WHERE ename IS NULL`

Logical operators

A logical operator combines the results of two component conditions to produce a single result or to invert the result of a single condition. The following table lists the supported logical operators.

Table 17: 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 18: Operator Precedence

Precedence	Operator
1	+ (Positive), - (Negative)
2	*(Multiply), / (Division)
3	+ (Add), - (Subtract)
4	|| (Concatenate)
5	=, >, <, >=, <=, <>, != (Comparison operators)
6	NOT, IN, LIKE

Precedence	Operator
7	AND
8	OR

Example A

The query in the following example returns employee records for which the department number is 1 or 2 and the salary is greater than $1000:

```
SELECT * FROM emp WHERE (deptno = 1 OR deptno = 2) AND sal > 1000
```

Because parenthetical expressions are forced to be evaluated first, the OR operation takes precedence over AND.

Example B

In the following example, the query returns records for all the employees in department 1, but only employees whose salary is greater than $1000 in department 2.

```
SELECT * FROM emp WHERE deptno = 1 OR deptno = 2 AND sal > 1000
```

The AND operator takes precedence over OR, so that the search condition in the example is equivalent to the expression `deptno = 1 OR (deptno = 2 AND sal > 1000)`.

Functions

The driver supports a number of functions that you can use in expressions, as listed and described in "Scalar functions."

Refer to "Scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for more information.

Conditions

A condition specifies a combination of one or more expressions and logical operators that evaluates to either TRUE, FALSE, or UNKNOWN. You can use a condition in the Where clause of the Delete, Select, and Update statements; and in the Having clauses of Select statements. The following describes supported conditions.

Table 19: Conditions

Condition	Description
Simple comparison	Specifies a comparison with expressions or subquery results. `= , !=, <>, < , >, <=, <=`
Group comparison	Specifies a comparison with any or all members in a list or subquery. `[= , !=, <>, < , >, <=, <=] [ANY, ALL, SOME]`

Condition	Description
Membership	Tests for membership in a list or subquery. `[NOT] IN`
Range	Tests for inclusion in a range. `[NOT] BETWEEN`
NULL	Tests for nulls. `IS NULL, IS NOT NULL`
EXISTS	Tests for existence of rows in a subquery. `[NOT] EXISTS`
LIKE	Specifies a test involving pattern matching. `[NOT] LIKE`
Compound	Specifies a combination of other conditions. `CONDITION [AND/OR] CONDITION`

7

Introduction to the SAP S/4HANA data model

The S/4HANA 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 OData API calls. To expose S/4HANA resources to SQL applications, the driver maps S/4HANA endpoints to one schema and each schema has a set of relational parent and child tables. The following sections describe the relational tables exposed by the driver along with their corresponding S/4HANA OData API call.

For details, see the following topics:

- A_BUSINESSPARTNERCONTACT
- A_BUSINESSPARTNERROLE
- A_BUSINESSPARTNERTAXNUMBER
- A_CUSTOMER
- A_CUSTOMERCOMPANY
- A_CUSTOMERDUNNING
- A_CUSTOMERSALESAREA
- A_CUSTOMERSALESAREATAX
- A_CUSTOMERWITHHOLDINGTAX
- A_CUSTSALESPARTNERFUNC
- A_SUPPLIER
- A_SUPPLIERCOMPANY
- A_SUPPLIERDUNNING
- A_SUPPLIERPARTNERFUNC
- A_SUPPLIERPURCHASINGORG
- A_SUPPLIERWITHHOLDINGTAX

A_ADDRESSEMAILADDRESS

Columns

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

Column Name	Data Type
ADDRESSID*	STRING(10)
PERSON*	STRING(10)
ORDINALNUMBER*	STRING(3)
EMAILADDRESS	STRING(241)
ISDEFAULTEMAILADDRESS	BOOLEAN
SEARCHEMAILADDRESS	STRING(20)

A_ADDRESSFAXNUMBER

Columns

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

Column Name	Data Type
ADDRESSID*	STRING(10)
PERSON*	STRING(10)
ORDINALNUMBER*	STRING(3)
FAXCOUNTRY	STRING(3)
FAXNUMBER	STRING(30)
FAXNUMBEREXTENSION	STRING(10)
INTERNATIONALFAXNUMBER	STRING(30)
ISDEFAULTFAXNUMBER	BOOLEAN

A_ADDRESSHOMEPAGEURL

Columns

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

Column Name	Data Type
ADDRESSID*	STRING(10)
PERSON*	STRING(10)
ORDINALNUMBER*	STRING(3)
VALIDITYSTARTDATE*	DATETIME(0)
ISDEFAULTURLADDRESS*	BOOLEAN
SEARCHURLADDRESS	STRING(50)

Column Name	Data Type
URLFIELDLENGTH	INT16
WEBSITEURL	STRING(2048)

A_ADDRESSPHONENUMBER

Columns

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

Column Name	Data Type
ADDRESSID*	STRING(10)
PERSON*	STRING(10)
ORDINALNUMBER*	STRING(3)
DESTINATIONLOCATIONCOUNTRY	STRING(3)
INTERNATIONALPHONENUMBER	STRING(30)
ISDEFAULTPHONENUMBER	BOOLEAN
PHONENUMBER	STRING(30)
PHONENUMBEREXTENSION	STRING(10)
PHONENUMBERTYPE	STRING(1)

A_BPCONTACTTOADDRESS

Columns

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

Column Name	Data Type
RELATIONSHIPNUMBER*	STRING(12)
BUSINESSPARTNERCOMPANY*	STRING(10)
BUSINESSPARTNERPERSON*	STRING(10)

Column Name	Data Type
VALIDITYENDDATE*	DATETIME(0)
ADDRESSID*	STRING(10)
ADDITIONALSTREETPREFIXNAME	STRING(40)
ADDITIONALSTREETSUFFIXNAME	STRING(40)
ADDRESSNUMBER	STRING(10)
ADDRESSTIMEZONE	STRING(6)
CAREOFNAME	STRING(40)
CITYCODE	STRING(12)
CITYNAME	STRING(40)
COMPANYPOSTALCODE	STRING(10)
COUNTRY	STRING(3)
COUNTY	STRING(40)
DELIVERYSERVICENUMBER	STRING(10)
DELIVERYSERVICETYPECODE	STRING(4)
DISTRICT	STRING(40)
FORMOFADDRESS	STRING(4)
FULLNAME	STRING(80)
HOMECITYNAME	STRING(40)
HOUSENUMBER	STRING(10)
HOUSENUMBERSUPPLEMENTTEXT	STRING(10)
LANGUAGE	STRING(2)
PERSON	STRING(10)
POBOX	STRING(10)
POBOXDEVIATINGCITYNAME	STRING(40)
POBOXDEVIATINGCOUNTRY	STRING(3)
POBOXDEVIATINGREGION	STRING(3)

Column Name	Data Type
POBOXISWITHOUTNUMBER	BOOLEAN
POBOXLOBBYNAME	STRING(40)
POBOXPOSTALCODE	STRING(10)
POSTALCODE	STRING(10)
PRFRDCOMMMEDIUMTYPE	STRING(3)
REGION	STRING(3)
STREETNAME	STRING(60)
STREETPREFIXNAME	STRING(40)
STREETSUFFIXNAME	STRING(40)
TAXJURISDICTION	STRING(15)
TRANSPORTZONE	STRING(10)

A_BPCONTACTTOFUNCANDDEPT

Columns

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

Column Name	Data Type
RELATIONSHIPNUMBER*	STRING(12)
BUSINESSPARTNERCOMPANY*	STRING(10)
BUSINESSPARTNERPERSON*	STRING(10)
VALIDITYENDDATE*	DATETIME(0)
CONTACTPERSONDEPARTMENT	STRING(4)
CONTACTPERSONFUNCTION	STRING(4)
EMAILADDRESS	STRING(241)
FAXNUMBER	STRING(30)
FAXNUMBEREXTENSION	STRING(10)

Column Name	Data Type
PHONENUMBER	STRING(30)
PHONENUMBEREXTENSION	STRING(10)
RELATIONSHIPCATEGORY	STRING(6)

A_BUPAADDRESSUSAGE

Columns

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

Column Name	Data Type
BUSINESSPARTNER*	STRING(10)
VALIDITYENDDATE*	DATETIME(0)
ADDRESSUSAGE*	STRING(10)
ADDRESSID*	STRING(10)
AUTHORIZATIONGROUP	STRING(4)
STANDARDUSAGE	BOOLEAN
VALIDITYSTARTDATE	DATETIME(0)

A_BUPAIDENTIFICATION

Columns

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

Column Name	Data Type
BUSINESSPARTNER*	STRING(10)
BPIDENTIFICATIONTYPE*	STRING(6)
BPIDENTIFICATIONNUMBER*	STRING(60)
AUTHORIZATIONGROUP	STRING(4)

Column Name	Data Type
BPIDENTIFICATIONENTRYDATE	DATETIME(0)
BPIDNNMBRISSUINGINSTITUTE	STRING(40)
COUNTRY	STRING(3)
REGION	STRING(3)
VALIDITYENDDATE	DATETIME(0)
VALIDITYSTARTDATE	DATETIME(0)

A_BUPAINDUSTRY

Columns

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

Column Name	Data Type
INDUSTRYSECTOR*	STRING(10)
INDUSTRYSYSTEMTYPE*	STRING(4)
BUSINESSPARTNER*	STRING(10)
INDUSTRYKEYDESCRIPTION	STRING(100)
ISSTANDARDINDUSTRY	STRING(1)

A_BUSINESSPARTNER

Columns

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

Column Name	Data Type
BUSINESSPARTNER*	STRING(10)
ACADEMICTITLE	STRING(4)
ADDITIONALLASTNAME	STRING(40)

Column Name	Data Type
AUTHORIZATIONGROUP	STRING(4)
BIRTHDATE	DATETIME(0)
BUSINESSPARTNERCATEGORY	STRING(1)
BUSINESSPARTNERFULLNAME	STRING(81)
BUSINESSPARTNERGROUPING	STRING(4)
BUSINESSPARTNERIDBYEXTSYSTEM	STRING(20)
BUSINESSPARTNERISBLOCKED	BOOLEAN
BUSINESSPARTNERNAME	STRING(81)
BUSINESSPARTNERTYPE	STRING(4)
BUSINESSPARTNERUUID	GUID(36)
CORRESPONDENCELANGUAGE	STRING(2)
CREATEDBYUSER	STRING(12)
CREATIONDATE	DATETIME(0)
CREATIONTIME	TIMEOFDAY(0)
CUSTOMER	STRING(10)
ETAG	STRING(26)
FIRSTNAME	STRING(40)
FORMOFADDRESS	STRING(4)
GROUPBUSINESSPARTNERNAME1	STRING(40)
GROUPBUSINESSPARTNERNAME2	STRING(40)
INDEPENDENTADDRESSID	STRING(10)
INDUSTRY	STRING(10)
INTERNATIONALLOCATIONNUMBER1	STRING(7)
INTERNATIONALLOCATIONNUMBER2	STRING(5)
INTERNATIONALLOCATIONNUMBER3	STRING(1)
ISFEMALE	BOOLEAN

Column Name	Data Type
ISMALE	BOOLEAN
ISMARKEDFORARCHIVING	BOOLEAN
ISNATURALPERSON	STRING(1)
ISSEXUNKNOWN	BOOLEAN
LANGUAGE	STRING(2)
LASTCHANGEDATE	DATETIME(0)
LASTCHANGEDBYUSER	STRING(12)
LASTCHANGETIME	TIMEOFDAY(0)
LASTNAME	STRING(40)
LEGALFORM	STRING(2)
MIDDLENAME	STRING(40)
NAMECOUNTRY	STRING(3)
NAMEFORMAT	STRING(2)
ORGANIZATIONBPNAME1	STRING(40)
ORGANIZATIONBPNAME2	STRING(40)
ORGANIZATIONBPNAME3	STRING(40)
ORGANIZATIONBPNAME4	STRING(40)
ORGANIZATIONFOUNDATIONDATE	DATETIME(0)
ORGANIZATIONLIQUIDATIONDATE	DATETIME(0)
PERSONFULLNAME	STRING(80)
PERSONNUMBER	STRING(10)
SEARCHTERM1	STRING(20)
SUPPLIER	STRING(10)
TRADINGPARTNER	STRING(6)

A_BUSINESSPARTNERADDRESS

Columns

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

Column Name	Data Type
BUSINESSPARTNER*	STRING(10)
ADDRESSID*	STRING(10)
ADDITIONALSTREETPREFIXNAME	STRING(40)
ADDITIONALSTREETSUFFIXNAME	STRING(40)
ADDRESSIDBYEXTERNALSYSTEM	STRING(20)
ADDRESSTIMEZONE	STRING(6)
ADDRESSUUID	GUID(36)
AUTHORIZATIONGROUP	STRING(4)
CAREOFNAME	STRING(40)
CITYCODE	STRING(12)
CITYNAME	STRING(40)
COMPANYPOSTALCODE	STRING(10)
COUNTRY	STRING(3)
COUNTY	STRING(40)
DELIVERYSERVICENUMBER	STRING(10)
DELIVERYSERVICETYPECODE	STRING(4)
DISTRICT	STRING(40)
FORMOFADDRESS	STRING(4)
FULLNAME	STRING(80)
HOMECITYNAME	STRING(40)
HOUSENUMBER	STRING(10)

Column Name	Data Type
HOUSENUMBERSUPPLEMENTTEXT	STRING(10)
LANGUAGE	STRING(2)
PERSON	STRING(10)
POBOX	STRING(10)
POBOXDEVIATINGCITYNAME	STRING(40)
POBOXDEVIATINGCOUNTRY	STRING(3)
POBOXDEVIATINGREGION	STRING(3)
POBOXISWITHOUTNUMBER	BOOLEAN
POBOXLOBBYNAME	STRING(40)
POBOXPOSTALCODE	STRING(10)
POSTALCODE	STRING(10)
PRFRDCOMMMEDIUMTYPE	STRING(3)
REGION	STRING(3)
STREETNAME	STRING(60)
STREETPREFIXNAME	STRING(40)
STREETSUFFIXNAME	STRING(40)
TAXJURISDICTION	STRING(15)
TRANSPORTZONE	STRING(10)
VALIDITYENDDATE	DATETIME(0)
VALIDITYSTARTDATE	DATETIME(0)

A_BUSINESSPARTNERBANK

Columns

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

Column Name	Data Type
BUSINESSPARTNER*	STRING(10)
BANKIDENTIFICATION*	STRING(4)
AUTHORIZATIONGROUP	STRING(4)
BANKACCOUNT	STRING(18)
BANKACCOUNTHOLDERNAME	STRING(60)
BANKACCOUNTNAME	STRING(40)
BANKACCOUNTREFERENCETEXT	STRING(20)
BANKCONTROLKEY	STRING(2)
BANKCOUNTRYKEY	STRING(3)
BANKNAME	STRING(60)
BANKNUMBER	STRING(15)
CITYNAME	STRING(35)
COLLECTIONAUTHIND	BOOLEAN
IBAN	STRING(34)
IBANVALIDITYSTARTDATE	DATETIME(0)
SWIFTCODE	STRING(11)
VALIDITYENDDATE	DATETIME(0)
VALIDITYSTARTDATE	DATETIME(0)

A_BUSINESSPARTNERCONTACT

Columns

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

Column Name	Data Type
RELATIONSHIPNUMBER*	STRING(12)
BUSINESSPARTNERCOMPANY*	STRING(10)

Column Name	Data Type
BUSINESSPARTNERPERSON*	STRING(10)
VALIDITYENDDATE*	DATETIME(0)
ISSTANDARDRELATIONSHIP	BOOLEAN
RELATIONSHIPCATEGORY	STRING(6)
VALIDITYSTARTDATE	DATETIME(0)

A_BUSINESSPARTNERROLE

Columns

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

Column Name	Data Type
BUSINESSPARTNER*	STRING(10)
BUSINESSPARTNERROLE*	STRING(6)
AUTHORIZATIONGROUP	STRING(4)
VALIDFROM	DATETIME(0)
VALIDTO	DATETIME(0)

A_BUSINESSPARTNERTAXNUMBER

Columns

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

Column Name	Data Type
BUSINESSPARTNER*	STRING(10)
BPTAXTYPE*	STRING(4)
AUTHORIZATIONGROUP	STRING(4)

Column Name	Data Type
BPTAXLONGNUMBER	STRING(60)
BPTAXNUMBER	STRING(20)

A_CUSTOMER

Columns

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

Column Name	Data Type
CUSTOMER*	STRING(10)
AUTHORIZATIONGROUP	STRING(4)
BILLINGISBLOCKEDFORCUSTOMER	STRING(2)
CREATEDBYUSER	STRING(12)
CREATIONDATE	DATETIME(0)
CUSTOMERACCOUNTGROUP	STRING(4)
CUSTOMERCLASSIFICATION	STRING(2)
CUSTOMERCORPORATEGROUP	STRING(10)
CUSTOMERFULLNAME	STRING(220)
CUSTOMERNAME	STRING(80)
DELETIONINDICATOR	BOOLEAN
DELIVERYISBLOCKED	STRING(2)
FISCALADDRESS	STRING(10)
INDUSTRY	STRING(4)
INDUSTRYCODE1	STRING(10)
INDUSTRYCODE2	STRING(10)
INDUSTRYCODE3	STRING(10)
INDUSTRYCODE4	STRING(10)

Column Name	Data Type
INDUSTRYCODE5	STRING(10)
INTERNATIONALLOCATIONNUMBER1	STRING(7)
NFPARTNERISNATURALPERSON	STRING(1)
NIELSENREGION	STRING(2)
ORDERISBLOCKEDFORCUSTOMER	STRING(2)
POSTINGISBLOCKED	BOOLEAN
RESPONSIBLETYPE	STRING(2)
SUPPLIER	STRING(10)
TAXNUMBER1	STRING(16)
TAXNUMBER2	STRING(11)
TAXNUMBER3	STRING(18)
TAXNUMBER4	STRING(18)
TAXNUMBER5	STRING(60)
TAXNUMBERTYPE	STRING(2)
VATREGISTRATION	STRING(20)

A_CUSTOMERCOMPANY

Columns

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

Column Name	Data Type
CUSTOMER*	STRING(10)
COMPANYCODE*	STRING(4)
ACCOUNTBYCUSTOMER	STRING(12)
ACCOUNTINGCLERK	STRING(2)
ACCOUNTINGCLERKFAXNUMBER	STRING(31)

Column Name	Data Type
ACCOUNTINGCLERKINTERNETADDRESS	STRING(130)
ACCOUNTINGCLERKPHONENUMBER	STRING(30)
ALTERNATIVEPAYERACCOUNT	STRING(10)
APARTOLERANCEGROUP	STRING(4)
AUTHORIZATIONGROUP	STRING(4)
COLLECTIVEINVOICEVARIANT	STRING(1)
CUSTOMERACCOUNTGROUP	STRING(4)
CUSTOMERACCOUNTNOTE	STRING(30)
CUSTOMERHEADOFFICE	STRING(10)
CUSTOMERSUPPLIERCLEARINGISUSED	BOOLEAN
DELETIONINDICATOR	BOOLEAN
HOUSEBANK	STRING(5)
INTERESTCALCULATIONCODE	STRING(2)
INTERESTCALCULATIONDATE	DATETIME(0)
INTRSTCALCFREQUENCYINMONTHS	STRING(2)
ISTOBELOCALLYPROCESSED	BOOLEAN
ITEMISTOBEPAIDSEPARATELY	BOOLEAN
LAYOUTSORTINGRULE	STRING(3)
PAYMENTBLOCKINGREASON	STRING(1)
PAYMENTMETHODSLIST	STRING(10)
PAYMENTTERMS	STRING(4)
PAYTADVICEISSENTBYEDI	BOOLEAN
PHYSICALINVENTORYBLOCKIND	BOOLEAN
RECONCILIATIONACCOUNT	STRING(10)
RECORDPAYMENTHISTORYINDICATOR	BOOLEAN
USERATCUSTOMER	STRING(15)

A_CUSTOMERDUNNING

Columns

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

Column Name	Data Type
CUSTOMER*	STRING(10)
COMPANYCODE*	STRING(4)
DUNNINGAREA*	STRING(2)
AUTHORIZATIONGROUP	STRING(4)
CUSTOMERACCOUNTGROUP	STRING(4)
DUNNINGBLOCK	STRING(1)
DUNNINGCLERK	STRING(2)
DUNNINGLEVEL	STRING(1)
DUNNINGPROCEDURE	STRING(4)
DUNNINGRECIPIENT	STRING(10)
LASTDUNNEDON	DATETIME(0)
LEGDUNNINGPROCEDUREON	DATETIME(0)

A_CUSTOMERSALESAREA

Columns

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

Column Name	Data Type
CUSTOMER*	STRING(10)
SALESORGANIZATION*	STRING(4)
DISTRIBUTIONCHANNEL*	STRING(2)

Column Name	Data Type
DIVISION*	STRING(2)
ACCOUNTBYCUSTOMER	STRING(12)
AUTHORIZATIONGROUP	STRING(4)
BILLINGISBLOCKEDFORCUSTOMER	STRING(2)
COMPLETEDELIVERYISDEFINED	BOOLEAN
CURRENCY	STRING(5)
CUSTOMERABCCLASSIFICATION	STRING(2)
CUSTOMERACCOUNTASSIGNMENTGROUP	STRING(2)
CUSTOMERACCOUNTGROUP	STRING(4)
CUSTOMERGROUP	STRING(2)
CUSTOMERPAYMENTTERMS	STRING(4)
CUSTOMERPRICEGROUP	STRING(2)
CUSTOMERPRICINGPROCEDURE	STRING(2)
DELETIONINDICATOR	BOOLEAN
DELIVERYISBLOCKEDFORCUSTOMER	STRING(2)
DELIVERYPRIORITY	STRING(2)
EXCHANGERATETYPE	STRING(4)
INCOTERMSCLASSIFICATION	STRING(3)
INCOTERMSLOCATION1	STRING(70)
INCOTERMSLOCATION2	STRING(70)
INCOTERMSTRANSFERLOCATION	STRING(28)
INCOTERMSVERSION	STRING(4)
INVOICEDATE	STRING(2)
INVOICELISTSCHEDULE	STRING(2)
ITEMORDERPROBABILITYINPERCENT	STRING(3)
ORDERCOMBINATIONISALLOWED	BOOLEAN

Column Name	Data Type
ORDERISBLOCKEDFORCUSTOMER	STRING(2)
PARTIALDELIVERYISALLOWED	STRING(1)
PRICELISTTYPE	STRING(2)
SALESDISTRICT	STRING(6)
SALESGROUP	STRING(3)
SALESOFFICE	STRING(4)
SHIPPINGCONDITION	STRING(2)
SUPPLYINGPLANT	STRING(4)

A_CUSTOMERSALESAREATAX

Columns

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

Column Name	Data Type
CUSTOMER*	STRING(10)
SALESORGANIZATION*	STRING(4)
DISTRIBUTIONCHANNEL*	STRING(2)
DIVISION*	STRING(2)
DEPARTURECOUNTRY*	STRING(3)
CUSTOMERTAXCATEGORY*	STRING(4)
CUSTOMERTAXCLASSIFICATION	STRING(1)

A_CUSTOMERWITHHOLDINGTAX

Columns

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

Column Name	Data Type
CUSTOMER*	STRING(10)
COMPANYCODE*	STRING(4)
WITHHOLDINGTAXTYPE*	STRING(2)
AUTHORIZATIONGROUP	STRING(4)
EXEMPTIONDATEBEGIN	DATETIME(0)
EXEMPTIONDATEEND	DATETIME(0)
EXEMPTIONREASON	STRING(2)
OBLIGATIONDATEBEGIN	DATETIME(0)
OBLIGATIONDATEEND	DATETIME(0)
WITHHOLDINGTAXAGENT	BOOLEAN
WITHHOLDINGTAXCERTIFICATE	STRING(25)
WITHHOLDINGTAXCODE	STRING(2)
WITHHOLDINGTAXEXMPTPERCENT	DECIMAL(5, 2)
WITHHOLDINGTAXNUMBER	STRING(16)

A_CUSTSALESPARTNERFUNC

Columns

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

Column Name	Data Type
CUSTOMER*	STRING(10)
SALESORGANIZATION*	STRING(4)
DISTRIBUTIONCHANNEL*	STRING(2)
DIVISION*	STRING(2)
PARTNERCOUNTER*	STRING(3)
PARTNERFUNCTION*	STRING(2)

Column Name	Data Type
AUTHORIZATIONGROUP	STRING(4)
BPCUSTOMERNUMBER	STRING(10)
CUSTOMERPARTNERDESCRIPTION	STRING(30)
DEFAULTPARTNER	BOOLEAN

A_SUPPLIER

Columns

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

Column Name	Data Type
SUPPLIER*	STRING(10)
ALTERNATIVEPAYEEACCOUNTNUMBER	STRING(10)
AUTHORIZATIONGROUP	STRING(4)
BIRTHDATE	DATETIME(0)
CONCATENATEDINTERNATIONALLOCNO	STRING(20)
CREATEDBYUSER	STRING(12)
CREATIONDATE	DATETIME(0)
CUSTOMER	STRING(10)
DELETIONINDICATOR	BOOLEAN
FISCALADDRESS	STRING(10)
INDUSTRY	STRING(4)
INTERNATIONALLOCATIONNUMBER1	STRING(7)
INTERNATIONALLOCATIONNUMBER2	STRING(5)
INTERNATIONALLOCATIONNUMBER3	STRING(1)
ISNATURALPERSON	STRING(1)
PAYMENTISBLOCKEDFORSUPPLIER	BOOLEAN

Column Name	Data Type
POSTINGISBLOCKED	BOOLEAN
PURCHASINGISBLOCKED	BOOLEAN
RESPONSIBLETYPE	STRING(2)
SUPLRPROOFOFDELIVRLVTCODE	STRING(1)
SUPLRQLTYINPROCMTCERTFNVALIDTO	DATETIME(0)
SUPLRQUALITYMANAGEMENTSYSTEM	STRING(4)
SUPPLIERACCOUNTGROUP	STRING(4)
SUPPLIERCORPORATEGROUP	STRING(10)
SUPPLIERFULLNAME	STRING(220)
SUPPLIERNAME	STRING(80)
SUPPLIERPROCUREMENTBLOCK	STRING(2)
TAXNUMBER1	STRING(16)
TAXNUMBER2	STRING(11)
TAXNUMBER3	STRING(18)
TAXNUMBER4	STRING(18)
TAXNUMBER5	STRING(60)
TAXNUMBERRESPONSIBLE	STRING(18)
TAXNUMBERTYPE	STRING(2)
VATREGISTRATION	STRING(20)

A_SUPPLIERCOMPANY

Columns

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

Column Name	Data Type
SUPPLIER*	STRING(10)

Column Name	Data Type
COMPANYCODE*	STRING(4)
ACCOUNTINGCLERK	STRING(2)
ACCOUNTINGCLERKFAXNUMBER	STRING(31)
ACCOUNTINGCLERKPHONENUMBER	STRING(30)
ALTERNATIVEPAYEE	STRING(10)
APARTOLERANCEGROUP	STRING(4)
AUTHORIZATIONGROUP	STRING(4)
BILLOFEXCHLMTAMTINCOCODECRCY	DECIMAL(14, 3)
CASHPLANNINGGROUP	STRING(10)
CHECKPAIDDURATIONINDAYS	DECIMAL(3, 0)
CLEARCUSTOMERSUPPLIER	BOOLEAN
COMPANYCODENAME	STRING(25)
CURRENCY	STRING(5)
DELETIONINDICATOR	BOOLEAN
HOUSEBANK	STRING(5)
INTERESTCALCULATIONCODE	STRING(2)
INTERESTCALCULATIONDATE	DATETIME(0)
INTRSTCALCFREQUENCYINMONTHS	STRING(2)
ISTOBECHECKEDFORDUPLICATES	BOOLEAN
ISTOBELOCALLYPROCESSED	BOOLEAN
ITEMISTOBEPAIDSEPARATELY	BOOLEAN
LAYOUTSORTINGRULE	STRING(3)
PAYMENTBLOCKINGREASON	STRING(1)
PAYMENTISTOBESENTBYEDI	BOOLEAN
PAYMENTMETHODSLIST	STRING(10)
PAYMENTTERMS	STRING(4)

Column Name	Data Type
RECONCILIATIONACCOUNT	STRING(10)
SUPPLIERACCOUNTGROUP	STRING(4)
SUPPLIERACCOUNTNOTE	STRING(30)
SUPPLIERCERTIFICATIONDATE	DATETIME(0)
SUPPLIERCLERK	STRING(15)
SUPPLIERCLERKIDBYSUPPLIER	STRING(12)
SUPPLIERCLERKURL	STRING(130)
SUPPLIERHEADOFFICE	STRING(10)
SUPPLIERISBLOCKEDFORPOSTING	BOOLEAN
WITHHOLDINGTAXCOUNTRY	STRING(3)

A_SUPPLIERDUNNING

Columns

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

Column Name	Data Type
SUPPLIER*	STRING(10)
COMPANYCODE*	STRING(4)
DUNNINGAREA*	STRING(2)
AUTHORIZATIONGROUP	STRING(4)
DUNNINGBLOCK	STRING(1)
DUNNINGCLERK	STRING(2)
DUNNINGLEVEL	STRING(1)
DUNNINGPROCEDURE	STRING(4)
DUNNINGRECIPIENT	STRING(10)
LASTDUNNEDON	DATETIME(0)

Column Name	Data Type
LEGDUNNINGPROCEDUREON	DATETIME(0)
SUPPLIERACCOUNTGROUP	STRING(4)

A_SUPPLIERPARTNERFUNC

Columns

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

Column Name	Data Type
SUPPLIER*	STRING(10)
PURCHASINGORGANIZATION*	STRING(4)
SUPPLIERSUBRANGE*	STRING(6)
PLANT*	STRING(4)
PARTNERFUNCTION*	STRING(2)
PARTNERCOUNTER*	STRING(3)
AUTHORIZATIONGROUP	STRING(4)
CREATEDBYUSER	STRING(12)
CREATIONDATE	DATETIME(0)
DEFAULTPARTNER	BOOLEAN
REFERENCESUPPLIER	STRING(10)

A_SUPPLIERPURCHASINGORG

Columns

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

Column Name	Data Type
SUPPLIER*	STRING(10)

Column Name	Data Type
PURCHASINGORGANIZATION*	STRING(4)
AUTHORIZATIONGROUP	STRING(4)
CALCULATIONSCHEMAGROUPCODE	STRING(2)
DELETIONINDICATOR	BOOLEAN
INCOTERMSCLASSIFICATION	STRING(3)
INCOTERMSLOCATION1	STRING(70)
INCOTERMSLOCATION2	STRING(70)
INCOTERMSTRANSFERLOCATION	STRING(28)
INCOTERMSVERSION	STRING(4)
INVOICEISGOODSRECEIPTBASED	BOOLEAN
MATERIALPLANNEDDELIVERYDURN	DECIMAL(3, 0)
MINIMUMORDERAMOUNT	DECIMAL(14, 3)
PAYMENTTERMS	STRING(4)
PRICINGDATECONTROL	STRING(1)
PURCHASEORDERCURRENCY	STRING(5)
PURCHASINGGROUP	STRING(3)
PURCHASINGISBLOCKEDFORSUPPLIER	BOOLEAN
PURORDAUTOGENERATIONISALLOWED	BOOLEAN
SHIPPINGCONDITION	STRING(2)
SUPPLIERABCCLASSIFICATIONCODE	STRING(1)
SUPPLIERACCOUNTGROUP	STRING(4)
SUPPLIERPHONENUMBER	STRING(16)
SUPPLIERRESPSALESPERSONNAME	STRING(30)

A_SUPPLIERWITHHOLDINGTAX

Columns

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

Column Name	Data Type
SUPPLIER*	STRING(10)
COMPANYCODE*	STRING(4)
WITHHOLDINGTAXTYPE*	STRING(2)
AUTHORIZATIONGROUP	STRING(4)
EXEMPTIONDATEBEGIN	DATETIME(0)
EXEMPTIONDATEEND	DATETIME(0)
EXEMPTIONREASON	STRING(2)
ISWITHHOLDINGTAXSUBJECT	BOOLEAN
RECIPIENTTYPE	STRING(2)
WITHHOLDINGTAXCERTIFICATE	STRING(25)
WITHHOLDINGTAXCODE	STRING(2)
WITHHOLDINGTAXEXMPTPERCENT	DECIMAL(5, 2)
WITHHOLDINGTAXNUMBER	STRING(16)