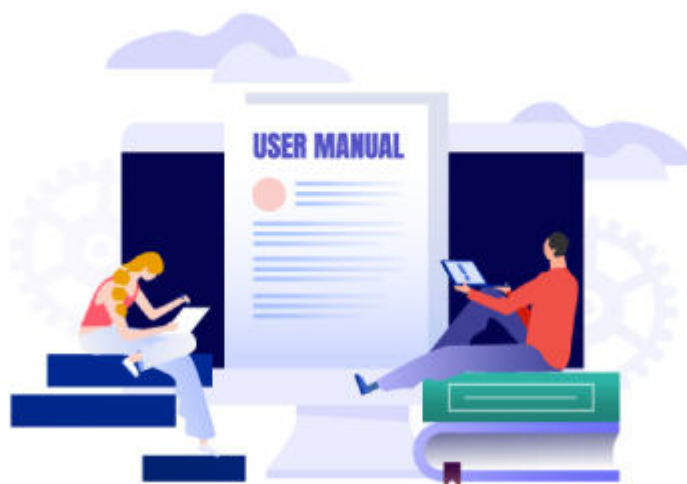


Unknown Title



Configuration of the ODBC Driver in Linux and Other UNIX¶

The Denodo Platform provides an ODBC driver for Linux, which is based on the ODBC PostgreSQL driver. There is one version for the driver manager unixODBC and one for iODBC.

You have to install the Denodo ODBC driver *in the machine where the client application runs*. To do this, follow these steps:

1. Obtain the appropriate ODBC driver
2. Install unixODBC
3. Register the ODBC driver with unixODBC
4. Register a data source (DSN) that points to Denodo

Obtain the Appropriate ODBC Driver¶

There are several flavors of the Denodo ODBC driver. This section explains which one you have to select:

1. Obtain the package `denodo-vdp-odbcdriver-linux.zip`. To do this:

1. Copy it from the installation (`<DENODO_HOME>/tools/client-drivers/odbc/denodo-vdp-odbcdriver-linux.zip`).

2. Or download it from the [ODBC page](#) of the Denodo Community.

On this page, download the driver for Linux (the name ends up with *-linux*). Make sure you select a version of the package that is *not newer* than the Denodo server you are going to connect. For example, if your Denodo server has the update 7.0 20181011, *do not* download the package *denodo-vdp-odbcdriver-7.0-update-20190312-linux* because it is newer.

The section [Access Through ODBC](#) explains the policy regarding backward compatibility of this driver.

2. Extract the contents of the file `denodo-vdp-odbcdriver-linux.tar.gz`:

```
tar -xzf denodo-vdp-odbcdriver-linux.tar.gz --directory /opt
```

This will create the folder `/opt/denodo-vdp-odbcdriver-linux`

3. Choose the appropriate flavor of the driver and copy all its files to the host where the client application runs. The options are:

- `unixodbc_x86`: ODBC driver for 32-bit clients and the unixODBC driver manager.
- `unixodbc_x64`: ODBC driver for 64-bit clients and the unixODBC driver manager.
- `iodbc_x86`: ODBC driver for 32-bit clients and the iODBC driver manager.
- `iodbc_x64`: ODBC driver for 64-bit clients and the iODBC driver manager.

For example, to connect from a 32-bit application using the unixODBC driver manager, copy the folder `unixodbc_x86` to the host where the client application runs.

Denodo also provides the ANSI version of each driver. These are the files ending with “a” (e.g. `unixodbc_x86/denodoodbca.so`). Only use the ANSI version when the Unicode encoding is not valid for your environment.

When loading the Denodo ODBC driver the following shared libraries (of a version compatible with the listed version) need to be accessible:

- `libc.so.6`: GLIBC_2.14 GLIBC_2.3.4 GLIBC_2.4 GLIBC_2.2.5
- `libpthread.so.0`: GLIBC_2.2.5
- `libdl.so.2`: GLIBC_2.2.5
- `libgssapi_krb5.so.2`: gssapi_krb5_2_MIT

Install UnixODBC¶

Linux does not provide an ODBC driver manager by default; you have to install it. This section explains how to install and configure [unixODBC](#). Denodo also provides the ODBC driver compiled to be used with iODBC.

Important

Install unixODBC and the Denodo ODBC driver on the host where the client application runs.

To verify if unixODBC is installed, execute the following commands. These check if the command line utilities `odbcinst` and `isql` are already installed:

```
which isql
```

```
which odbcinst
```

If both commands return the path to a file, go to the next section.

If unixODBC is not installed, do this:

1. For Linux distributions that use the RPM packaging system (e.g. Red Hat, CentOS, etc.), execute this:

```
sudo yum install unixODBC
```

For the ones based on Debian like Ubuntu execute this:

```
sudo apt-get install unixodbc
```

2. Execute this to verify that unixODBC was installed correctly:

```
odbcinst -j
```

Register the Denodo ODBC Driver in UnixODBC¶

After installing unixODBC, register the ODBC driver. Follow these steps:

1. Create a file `denodoODBCDriver.template` with this content:

```
1 [DenodoODBCDriver]
2 Description=ODBC driver of Denodo
3 Driver=/opt/denodo-odbc-driver/unixodbc_x86/denodoodbc.so
4 UsageCount=1
```

Modify line #3 so it points to the flavor of the ODBC driver you want to use (i.e. `unixodbc_x86`, `unixodbc_x64`,...).

2. Execute the following command to register the Denodo driver in the ODBC Driver Manager:

```
sudo odbcinst -install -driver -file denodoODBCDriver.template
```

To list the ODBC drivers registered in the driver manager, execute this:

```
sudo odbcinst -query -driver
```

The result should list the new driver: DenodoODBCDriver.

To uninstall the driver, execute:

```
sudo odbcinst -uninstall -driver -name DenodoODBCDriver
```

Register a Data Source (DSN) on UnixODBC¶

This section explains how to register a DSN in unixODBC.

1. Create a file called `denodoDSN.template` with the content below:

```
1 [Denodo_DSN]
2 Description = Denodo connection
3 Driver = DenodoODBCDriver
4 Servername = <host name>
5     # Default port of the ODBC interface of Virtual DataPort
6 Port = 9996
7 UserName = <Virtual DataPort user name>
8 Password = <Password>
9 Database = <Virtual DataPort database>
10 UserAgent = <name of the application that will use the DSN>
11 Protocol = 7.4
12 BoolAsChar = 0
13 ByteAsLongVarBinary= 1
14 ConnSettings = SET QUERYTIMEOUT TO 3600000; SET I18N TO us_pst;
15 /*krbsrvname=HTTP*/
16 Debug = 0
17 Commlog = 0
18 FakeOidIndex = 0
19 Fetch = 1000
20 Ksqo = 0
21 LFConversion = 1
22 Optimizer = 0
23 ReadOnly = 0
24 RowVersioning = 0
25 ShowOidColumn = 0
26 ShowSystemTables = 0
27 # Uncomment "Sslmode" if SSL is enabled in Virtual DataPort
28 # Sslmode = require
29 UniqueIndex = 1
30 UpdatableCursors = 0
31 UseDeclareFetch = 1
32 UseServerSidePrepare= 0
33 UseKerberos = 0
34 KeepaliveTime=60
35 KeepaliveInterval=5
36 InfrastructureProvider = <Name of infrastructure provider>
   InfrastructureRegion = <Name of infrastructure region>
```

In the line #7 (Password), some tools may ask you for a DSN name and then ask you for an user and password instead of using those defined in the DSN. you may not be able to connect if the password contains the following four characters: %, {, } and +. If so, you may need to escape these characters: % as %25, + as %2B, { as %7B and } as %7D (as alternative, you could set 'DecodeConnectionStringSecrets=0').

In the line #8 (`Database`), if the name of the database contains non-ASCII characters, they have to be URL-encoded. For example, if the name of the database is “テスト”, set the property to `%E3%83%86%E3%82%B9%E3%83%88`.

In the line #30 (`UseDeclareFetch`), if the value is 1, the DSN will use `DECLARE CURSOR/FETCH` to handle `SELECT` statements. The effect is that the DSN will retrieve the rows of the result set in blocks, instead of retrieving them all at once. The `Fetch` property establishes the number of rows of each block. This property is equivalent to the “Fetch size” of the JDBC connections.

In the lines #14 and #15 (`Debug` and `Commlog`), if the value is 1, the driver logs detailed ODBC information in files created in the `/tmp` directory. On a production environment, we strongly recommend setting the value of this property to 0 because logging all the requests impacts the performance of the driver and the log files may grow to a very large size.

To store these files on a different directory, create a `DenodoODBC Unicode` or `DenodoODBC ANSI` section in the `odbcinst.ini` file and define this property:

```
Logdir = /path/to/logs
```

In the line #33 and #34 there are two options related to TCP keepalive settings. The section [Set Up a DSN on Windows](#) explains these options.

In the line #35 and #36 there are two options related to specify the provider and region from a cloud provider. To know how they work, see the section [Setting Cloud Infrastructure Properties](#).

In the line #13 (`ConnSettings`), you can set the properties of the connection established with Virtual DataPort, by adding the following statements:

1. `SET QUERYTIMEOUT TO <value>` to change the query time out (value in milliseconds).
2. `SET i18n TO <i18n>` to change the i18n of the connection.

For example, to set the default timeout of the queries to one hour, set the value of the property `ConnSettings` to the following:

```
ConnSettings=SET QUERYTIMEOUT TO 3600000; SET I18N TO us_pst
```

Note the `;` between each statement.

Read [Parameters of the ODBC driver and their default value](#) to learn how these properties work, and their default value.

If you have enabled SSL in the Virtual DataPort server to secure the communications, add the following property to this configuration file:

Sslmode=require

3. Add the following `ConnSettings` property to connect to Virtual DataPort using Kerberos authentication:

```
/*krbsrvname=HTTP*/
```

Important

This line has to be the last thing on the `ConnSettings` property.

If Kerberos authentication is enabled on the Denodo database you are connecting to, the driver will ignore the value of the properties “UserName” and “Password”. Instead, it will obtain a Kerberos ticket from the system cache. Besides that, it is no longer required to enable Kerberos authentication at database level for ODBC connections; if you are using the latest Denodo ODBC Driver, you can set the property `UseKerberos` to 1 to enable Kerberos authentication at client side.

To be able to use Kerberos authentication, the configuration of the DSN has to meet these conditions:

1. The client has to belong to the Windows domain. The reason is that the ODBC driver uses the ticket cache of the operating system to obtain “ticket-granting ticket” (TGT).
2. In the property `Servername`, enter the fully qualified domain name of the Denodo server. That is, if in the Kerberos configuration of the Denodo server the field *Server principal* is `HTTP/denodo-prod.subnet1.contoso.com@CONTOSO.COM`, enter `denodo-prod.subnet1.contoso.com`.
4. The property `UserAgent` is optional but we recommend adding it to all the DSNs. That is because the user agent allows you to identify what application opens each connection and the requests that each application sends. This is useful for debugging problems caused by a particular client or for logging purposes.
5. To establish the connection using OAuth authentication instead of user and password, you will need to provide some of the following parameters depending on the desired OAuth authentication flow:

Parameters for OAuth authentication

Property	Meaning
UseOAuth2	Set to 1. Instructs the driver to open the connection with OAuth authentication.
TokenEndpoint	URL exposed by the OAuth server and used to request the <code>access_token</code> . For instance <code>https://login.microsoftonline.com/common/oauth2/token.</code>

Property	Meaning
AuthEndpoint	URL exposed by the OAuth server and used to request the authorization code. For instance <code>https://login.microsoftonline.com/common/oauth2/authorize.</code>
ClientId	Application's Client ID. Usually, you obtain this when registering the client application in the Identity Provider.
ClientSecret	Application's Client secret.
Scope	Space-delimited list of requested scope permissions.
ExtraParams	Additional parameters that will be added to the body of the HTTP requests the driver will send to obtain OAuth tokens. The syntax of the value of this parameter is <code>param1=value1&param2=value2&...</code>
AccessToken	The access token to be used to connect into Virtual DataPort server.
RefreshToken	Refresh token to get a new access token if the provided one is expired.
OAuthServiceTimeout	The timeout duration, in seconds, that the service responsible for obtaining an authorization code will wait for a response. It specifies the maximum time the service will remain idle while awaiting a reply during the authorization process. The default value is 120.
OAuthServiceMinPort	This parameter specifies the lower bound of the port range to be used by the service responsible for obtaining an authorization code. The default value is 8000.
OAuthServiceMaxPort	This parameter specifies the upper bound of the port range to be used by the service responsible for obtaining an authorization code. The default value is 9000.
OAuthCredentialsCacheFilePath	File path to a file where the tokens will be stored to avoid perform unnecessary requests to the identity provider. The <code>@{HOME}</code> interpolation variable can be used to reference user's home directory.
UseIdToken	If 1, the driver will use the "id_token" for authentication. If 0, it will use "access_token". Set this to 1 in an environment with "OpenID Connect".
OAuthSSLVerify	If 1, the driver will validate the SSL certificate of the Identity Provider. If 0, it will not validate the certificate. Default value: 1.

Currently, the ODBC driver supports three OAuth authentication flows and different parameters must be set for each one of them, be sure to select the appropriate ones for your scenario.

1. Resource owner credentials flow (ROPC). When the client application opens a connection, the driver will request an OAuth token to the Identity Provider and will use that token to establish the connection.
 - User and password.
 - TokenEndpoint.
 - ClientId.

- ClientSecret.
- Scope.
- ExtraParams (Optional).
- UseIdToken.
- OAuthSSLVerify.

2. Refresh token flow. The driver will use the provided access token to establish the connection. If the token is expired the driver will request a new OAuth token to the Identity Provider using the refresh token.

- TokenEndpoint.
- ClientId.
- ClientSecret.
- Scope.
- ExtraParams (Optional).
- Access token.
- Refresh token.
- OAuthCredentialsCacheFilePath (Optional).
- UseIdToken.
- OAuthSSLVerify.

3. Authorization code flow. The driver will request an access token to the Identity Provider using a web browser. It is highly recommended to set a credentials cache file to avoid open unnecessary tabs in the web browser when the client application implements some kind of connection pool.

- Authorization Endpoint.
- TokenEndpoint.
- ClientId.
- ClientSecret.
- Scope.
- ExtraParams (Optional).
- OAuthCredentialsCacheFilePath (Optional).

- OAuthServiceTimeout (Optional).
- OAuthServiceMinPort (Optional).
- OAuthServiceMaxPort (Optional).
- UseIdToken.
- OAuthSSLVerify.

Before using OAuth to connect to Virtual DataPort, you have to [enable OAuth in Virtual DataPort](#). Otherwise, the connections with OAuth will fail.

2. Execute this to register the new DSN:

```
sudo odbcinst -install -s -l -f denodoDSN.template
```

The parameter `-l` registers the DSN as a “system DSN”. “System DSNs” are available to all the users.

If you do not have enough privileges to register a “system DSN”, replace `-l` with `-h` to register the DSN as a “user DSN” instead. If you do this, execute this command with the same user name that you execute the client application that needs to access to this DSN. The reason is that “user DSNs” are only available to the user that registers them.

To list the DSNs registered in the ODBC driver manager, execute this:

```
sudo odbcinst -query -s
```

The result should list the new DSN: `Denodo_DSN`.

3. Execute this to test the DSN using the command line utility “isql” included with unixODBC:

```
isql -v Denodo_DSN
```

You should see something like this:

```
+-----+
| Connected!                                |
|                                           |
| sql-statement                            |
| help [tablename]                         |
| quit                                     |
|                                           |
+-----+
```

Execute any query (for example `SELECT 1`) and then, type `quit` to exit this shell.

After setting up the DSN, we recommend reading the section [Integration with Third-Party Applications](#).

Compiling UnixODBC¶

If you cannot install unixODBC using the package manager of your operating system, download it and compile it. To do this, follow these steps:

1. Download the latest version of the source code from <http://www.unixodbc.org/download.html>.
2. Execute the following commands to extract the source code and compile it:

```
tar -zxf unixODBC*.tar.gz
cd unixODBC
./configure.sh
make
```

3. Execute the following command:

```
sudo make install
```

Troubleshooting Issues¶

If your application returns an error like this one when trying to use this DSN:

```
[unixODBC][Driver Manager]Data source name not found, and no default driver
specified (0) (SQLDriverConnect)
```

follow these steps:

1. Connect to the host where you created the DSN.
2. Check that these files exist:

```
/usr/local/lib64/libodbc.so
/usr/local/lib64/libodbcinst.so
```

Or,

```
/usr/local/lib/libodbc.so
/usr/local/lib/libodbcinst.so
```

Their location may change depending on the Linux/Unix distribution.

3. Edit the file `~/.bash_profile` and add the following at the end:

```
export
LD_PRELOAD=/usr/local/lib/libodbc.so:/usr/local/lib/libodbcinst.so:$LD_PRELOAD
```

With this change in the value of the variable `LD_PRELOAD`, you make sure that the application loads the files `libodbc.so` and `libodbcinst.so` provided by unixODBC and not the ones provided by other libraries.

4. Logout and login again from this user account. Do this to apply the changes done in the file `.bash_profile`.

Note

If the two files listed above are in `lib64` and not in `lib`, change the line above accordingly.