EDB Postgres Advanced Server ODBC Connector Guide

Version 12.2.0.2

Built at 2021-01-19T10:46:51

1 What's New

The following features are added to create the EDB ODBC Connector 12.02.0000.02:

- Support for EDB Postgres Advanced Server 13.
- Support for Ubuntu 20.04 LTS platform.

2 Requirements Overview

Supported Versions

supported server versions

The EDB ODBC Connector is certified with Advanced Server version 9.5 and above.

Supported Platforms

The EDB ODBC Connector native packages are supported on the following platforms:

64 bit Linux:

- Red Hat Enterprise Linux (x86_64) 8.x and 7.x
- CentOS (x86_64) 8.x and 7.x
- OEL Linux 8.x and 7.x
- PPC-LE 8 running RHEL or CentOS 7.x
- SLES 12.x
- Debian 10.x and 9.x
- Ubuntu 20.04 and 18.04 LTS

The EDB ODBC Connector graphical installers are supported on the following Windows platforms:

64-bit Windows:

- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2
- Windows 10
- Windows 8.1

32-bit Windows:

- Windows 10
- Windows 8.1

3 EDB-ODBC Overview

EDB ODBC is an interface that allows an ODBC compliant client application to connect to an Advanced Server database. The EDB-ODBC connector allows an application that was designed to work with other databases to run on Advanced Server; EDB ODBC provides a way for the client application to establish a connection, send queries and retrieve results from Advanced Server.

While EDB ODBC provides a level of application portability, it should be noted that the portability is limited; EDB ODBC provides a connection, but does not guarantee command compatibility. Commands that are acceptable in another database, may not work in Advanced Server.

The major components in a typical ODBC application are:

- The client application - written in a language that has a binding for ODBC
- The ODBC Administrator - handles named connections for Windows or Linux
- The database specific ODBC driver - EDB ODBC
- The ODBC compliant server - EDB Postgres Advanced Server

Client applications can be written in any language that has a binding for ODBC; C, MS-Access, and C++ are just a few.

3.1 Installing EDB-ODBC

The EDB ODBC Connector is distributed and installed with the EDB Postgres Advanced Server graphical or RPM installer.

Installing the Connector with an RPM Package

Installing ODBC using RPM

You can install the ODBC Connector using an RPM package on the following platforms:

- RHEL 7
- RHEL 8
- CentOS 7
- CentOS 8

On RHEL 7

Before installing the ODBC Connector, you must install the following prerequisite packages, and request credentials from EDB:

Install the `epel-release` package:

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Enable the optional, extras, and HA repositories:

```
subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-extras-rpms"  --enable "rhel-ha-for-rhel-*-server-rpms"
```

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

> https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-odbc`.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

Modifying the file, providing your user name and password

After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Installing ODBC Connector

After saving your changes to the configuration file, use the following commands to install the ODBC Connector:

```
yum install edb-odbc

yum install edb-odbc-devel
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually

resolve.

On RHEL 8

rhel8

Before installing the ODBC Connector, you must install the following prerequisite packages, and request credentials from EDB:

Install the `epel-release` package:

```
dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

Enable the `codeready-builder-for-rhel-8-\*-rpms` repository:

```
ARCH=$( /bin/arch )
subscription-manager repos --enable "codeready-builder-for-rhel-8-${ARCH}-rpms"
```

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

> https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-odbc`.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
dnf -y https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

Modifying the file, providing your user name and password

After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Installing ODBC Connector

After saving your changes to the configuration file, use the below command to install the ODBC Connector:

```
dnf install edb-odbc

dnf install edb-odbc-devel
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

On CentOS 7

centos7

Before installing the ODBC Connector, you must install the following prerequisite packages, and request credentials from EDB:

Install the `epel-release` package:

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
7.noarch.rpm
```

Note

You may need to enable the `[extras]` repository definition in the `CentOS-Base.repo` file (located in `/etc/yum.repos.d`).

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

> https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-odbc`.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

Modifying the file, providing your user name and password

After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Installing ODBC Connector

After saving your changes to the configuration file, use the following command to

install the ODBC Connector:

```
yum install edb-odbc

yum install edb-odbc-devel
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

On CentOS 8

centos8

Before installing the ODBC Connector, you must install the following prerequisite packages, and request credentials from EDB:

Install the `epel-release` package:

```
dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

Enable the `PowerTools` repository:

```
dnf config-manager --set-enabled PowerTools
```

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

> https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-odbc`.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
dnf -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

Modifying the file, providing your user name and password

After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Installing ODBC Connector

After saving your changes to the configuration file, use the following command to install the ODBC Connector:

```
dnf install edb-odbc

dnf install edb-odbc-devel
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

Updating an RPM Installation

If you have an existing EDB ODBC connector RPM installation, you can use yum or dnf to upgrade your repository configuration file and update to a more recent product version. To update the `edb.repo` file, assume superuser privileges and enter:

- On RHEL or CentOS 7:

```
yum upgrade edb-repo
```

- On RHEL or CentOS 8:

```
dnf upgrade edb-repo
```

yum or dnf will update the `edb.repo` file to enable access to the current EDB repository, configured to connect with the credentials specified in your `edb.repo` file. Then, you can use yum or dnf to upgrade any installed packages:

- On RHEL or CentOS 7:

```
yum upgrade edb-odbc
```

```
yum upgrade edb-odbc-devel
```

- On RHEL or CentOS 8:

```
dnf upgrade edb-odbc
```

```
dnf upgrade edb-odbc-devel
```

Installing the Connector on an SLES 12 Host

Installing ODBC on SLES

You can use the zypper package manager to install the connector on an SLES 12 host. zypper will attempt to satisfy package dependencies as it installs a package, but requires access to specific repositories that are not hosted at EDB. Before installing the connector, use the following commands to add EDB repository configuration files to your SLES host:

```
zypper addrepo https://zypp.enterprisedb.com/suse/edb-sles.repo
```

After creating the repository configuration files, use the `zypper refresh` command to refresh the metadata on your SLES host to include the EDB repositories.

When prompted for a `User Name` and `Password`, provide your connection credentials for the EDB repository. To request credentials for the repository, visit the EDB website.

Before installing EDB Postgres Advanced Server or supporting components, you must also add SUSEConnect and the SUSE Package Hub extension to the SLES host, and register the host with SUSE, allowing access to SUSE repositories. Use the commands:

```
zypper install SUSEConnect
SUSEConnect -r 'REGISTRATION  CODE' -e 'EMAIL'
SUSEConnect -p PackageHub/12.4/x86_64
SUSEConnect -p sle-sdk/12.4/x86_64
```

For detailed information about registering a SUSE host, visit the SUSE website.

Then, you can use the zypper utility to install the connector:

```
zypper install edb-odbc
```

```
zypper install edb-odbc-devel
```

Installing the Connector on a Debian or Ubuntu Host

Installing ODBC on Debian

Installing ODBC on Ubuntu

To install a DEB package on a Debian or Ubuntu host, you must have credentials that allow access to the EDB repository. To request credentials for the repository, visit the EDB website.

The following steps will walk you through on using the EDB apt repository to install a DEB package. When using the commands, replace the `username` and `password` with the credentials provided by EDB.

1. Assume superuser privileges:

   ```
   sudo su –
   ```

2. Configure the EDB repository:

   On Debian 9:

   ```
   sh -c 'echo "deb
   ```

```
https://username:password@apt.enterprisedb.com/$(lsb_release -cs)-edb/
$(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -cs).list'
```

On Debian 10:

1. Set up the EDB repository:

```
-c 'echo "deb [arch=amd64] https://apt.enterprisedb.com/$(lsb_release -cs)-
edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -
cs).list'
```

1. Substitute your EDB credentials for the `username` and `password` in the following command:

```
-c 'echo "machine apt.enterprisedb.com login <username> password
<password>" > /etc/apt/auth.conf.d/edb.conf'
```

3. Add support to your system for secure APT repositories:

```
apt-get install apt-transport-https
```

4. Add the EDB signing key:

```
wget -q -O - https://<username>:<password>@apt.enterprisedb.com/edb-
deb.gpg.key | apt-key add -
```

5. Update the repository metadata:

```
apt-get update
```

6. Install DEB package:

```
apt-get install edb-odbc
apt-get install edb-odbc-dev
```

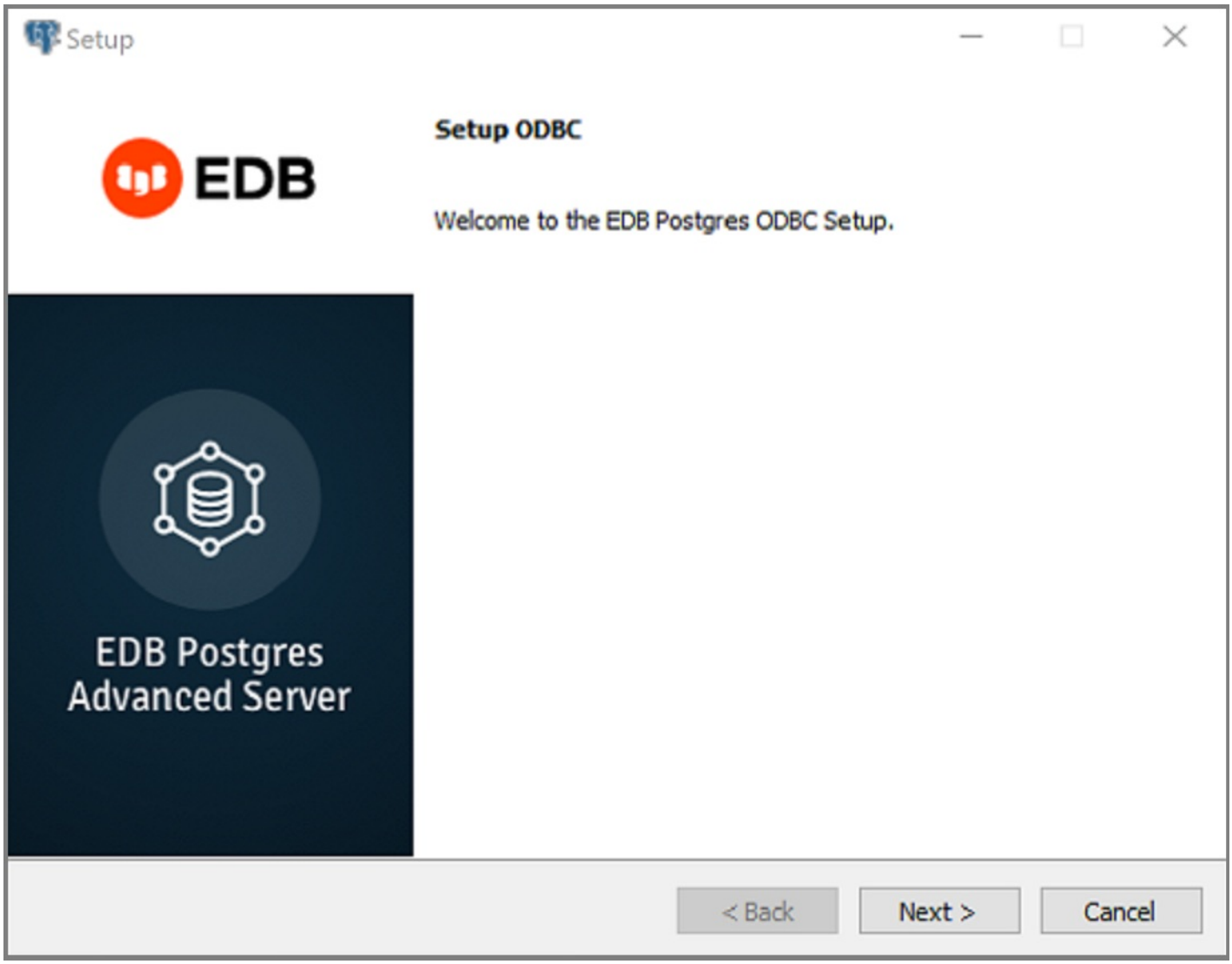
Using the Graphical Installer to Install the Connector

Installing ODBC with graphical installer

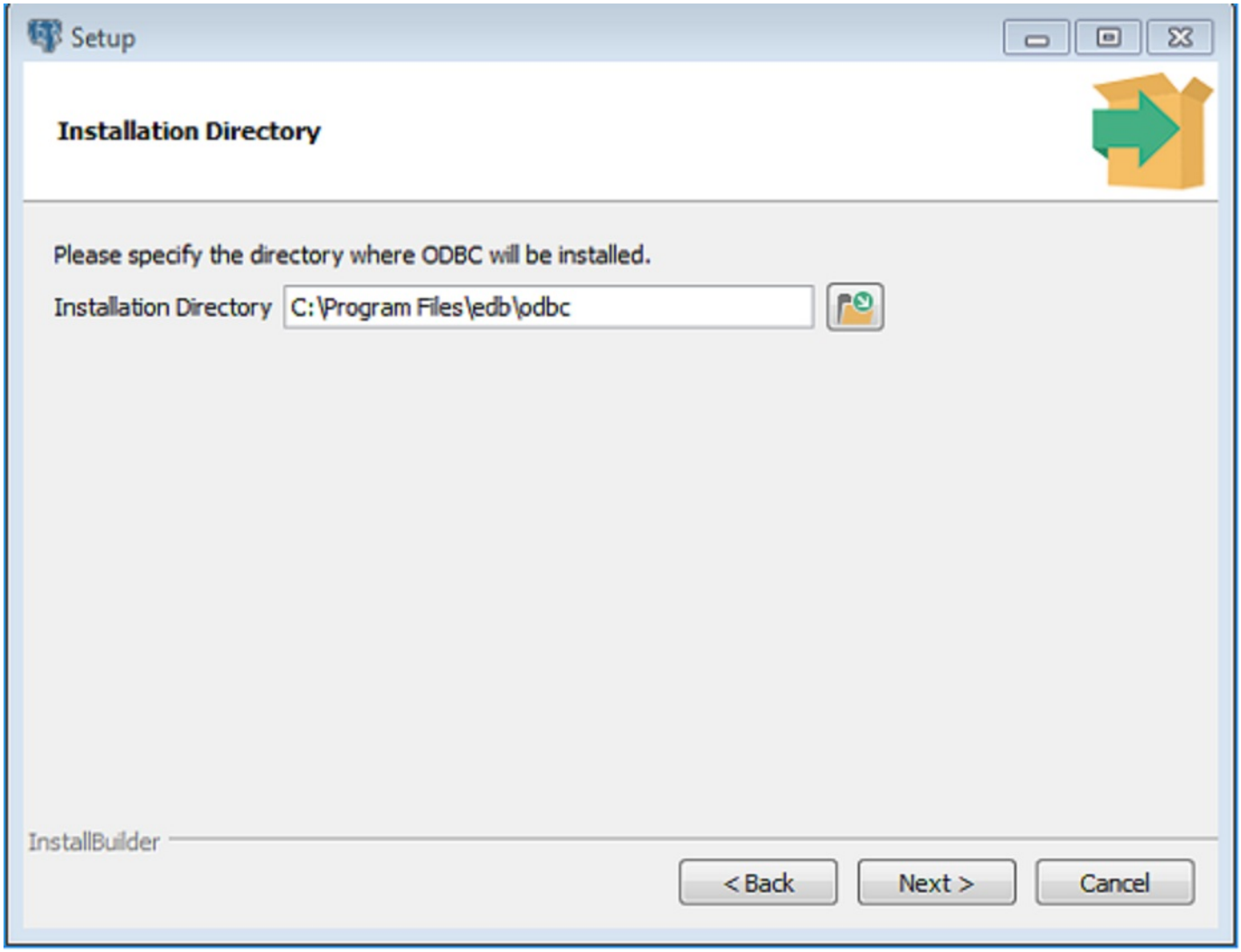
You can use the EDB Connectors Installation wizard to add the ODBC connector to your system; the wizard is available at the EDB website.

Download the installer, and then, right-click on the installer icon, and select Run As Administrator from the context menu.

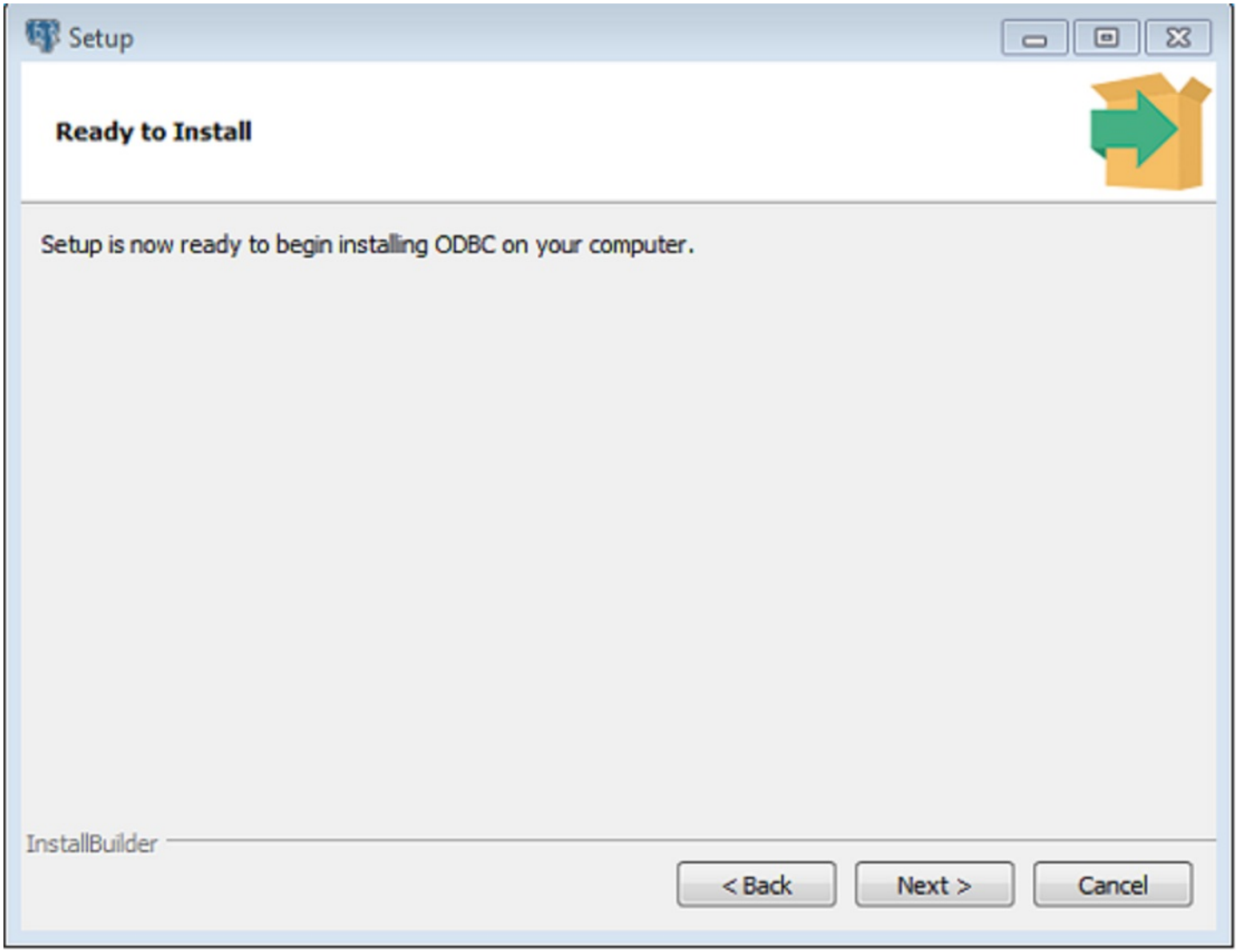
When the Language Selection popup opens, select an installation language and click OK to continue to the Setup window (shown in Figure below).



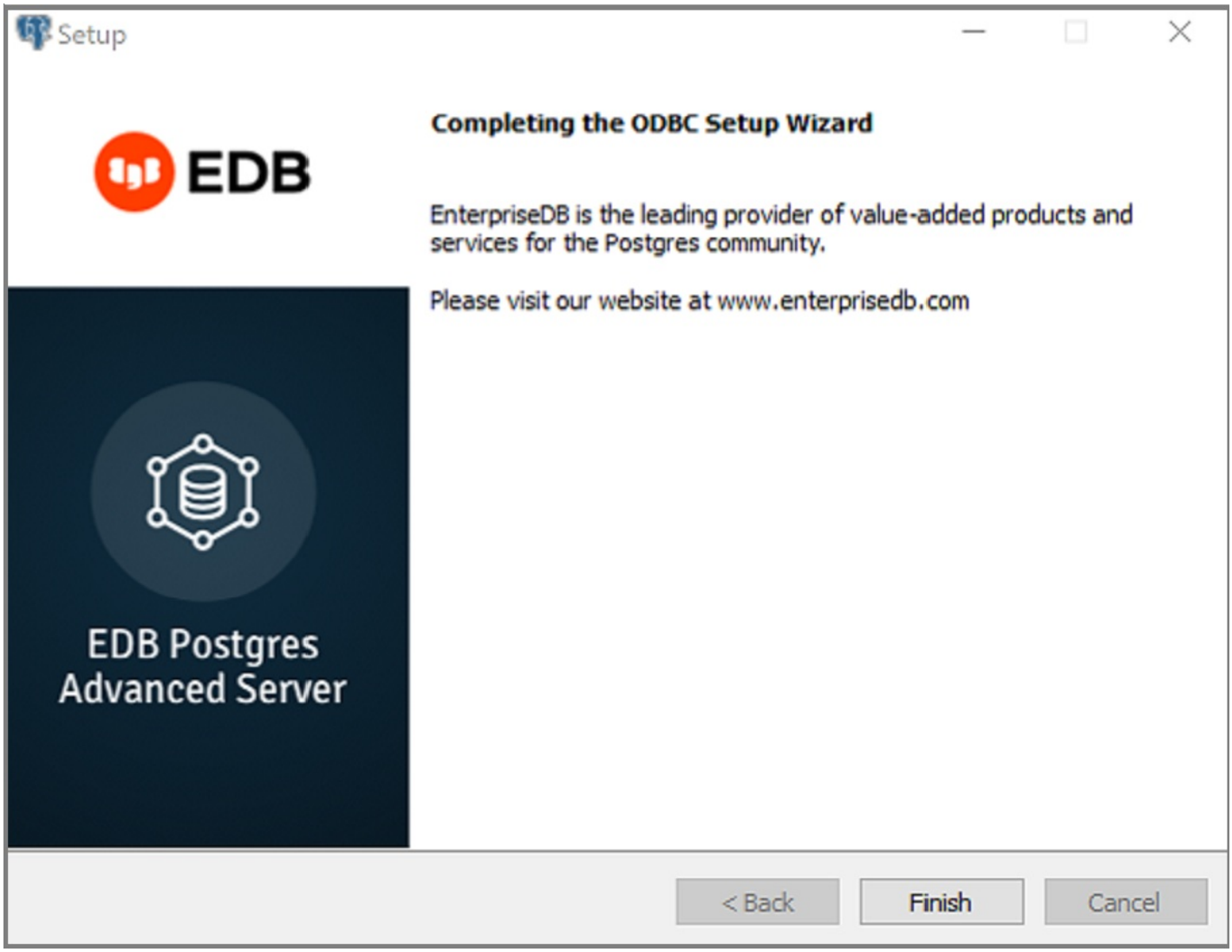
Click Next to continue.



Use the Installation Directory dialog to specify the directory in which the connector will be installed, and click Next to continue.

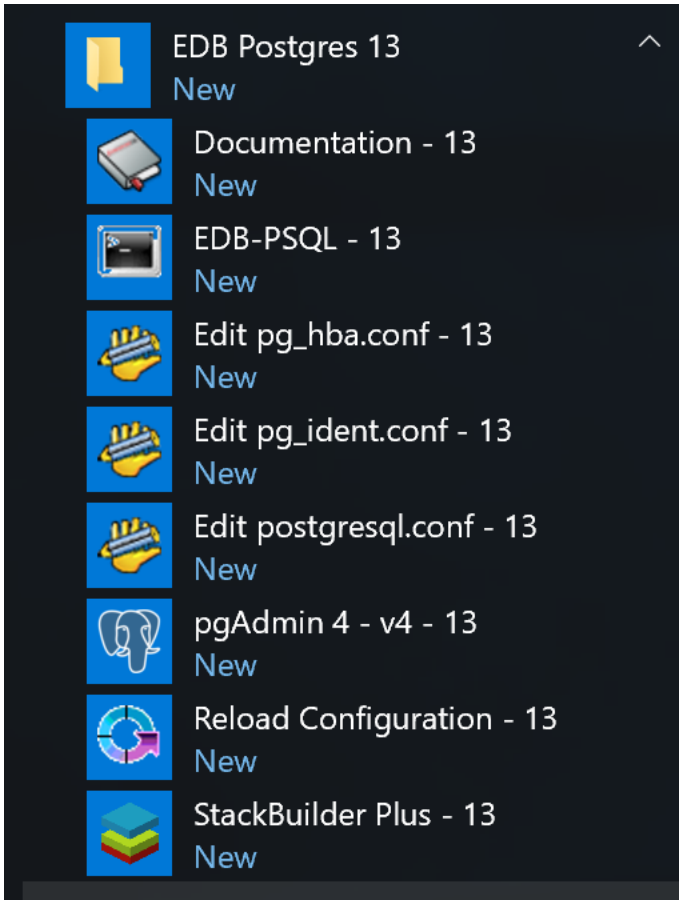


Click Next on the Ready to Install dialog to start the installation; popup dialogs confirm the progress of the installation wizard.

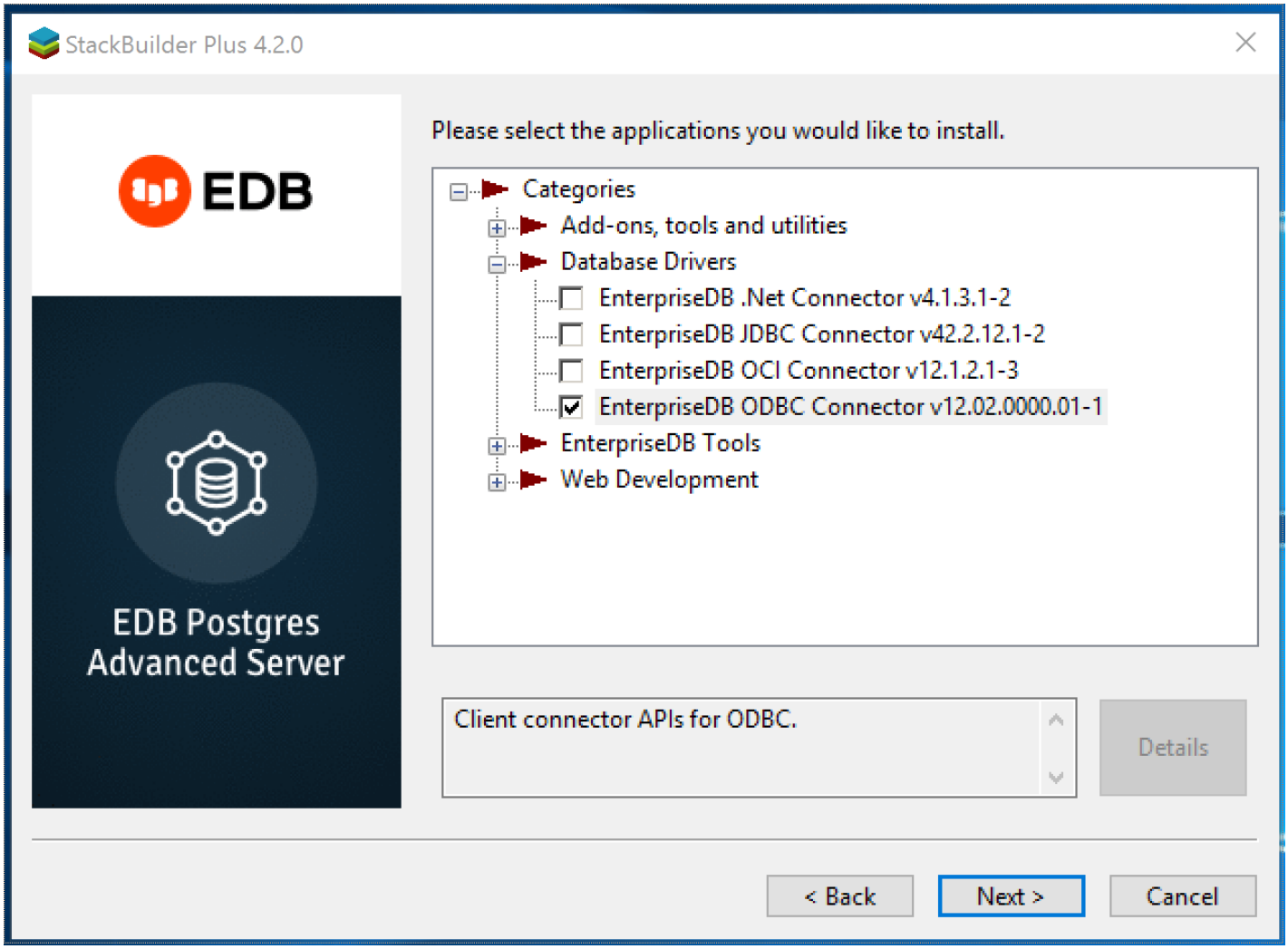


When the wizard informs you that it has completed the setup, click the Finish button to exit the dialog.

You can also use StackBuilder Plus to add or update the connector on an existing Advanced Server installation; to open StackBuilder Plus, select StackBuilder Plus from the Windows Apps menu.



When StackBuilder Plus opens, follow the onscreen instructions. Select the `EnterpriseDB ODBC Connector` option from the `Database Drivers` node of the tree control.



Follow the directions of the onscreen wizard to add or update an installation of the EDB Connectors.

4 Creating a Data Source

When a client application tries to establish a connection with a server, it typically provides a data source name (also known as a "DSN"). The driver manager looks through the ODBC configuration database for a data source whose name matches the DSN provided by the application.

On a Linux or Unix host, data sources are defined in a file; that file is usually named /etc/odbc.ini, but the name (and location) may vary. Use the following command to find out where unixODBC is searching for data source definitions:

```
$ odbc_config --odbcini --odbcinstini
```

On a Windows host, data sources are typically defined in the Windows registry.

You can also store a data source definition (called a "File DSN") in a plain-text file of your choice. A typical data source definition for the EDB-ODBC driver looks like this:

```
$ cat /etc/odbc.ini
[EnterpriseDB]
Description = EnterpriseDB DSN
Driver = EnterpriseDB
Trace = yes
TraceFile = /tmp/odbc.log
Database = edb
Servername = localhost
UserName = enterprisedb
Password = manager
Port = 5444
```

The first line in the data source is the data source name. The name is a unique identifier, enclosed in square brackets. The data source name is followed by a series of `'keyword=value'` pairs that identify individual connection properties that make up the data source.

The ODBC administrator utility creates named data sources for ODBC connections. In most cases, an ODBC administrator utility is distributed with the operating system (if you're using Windows or unixODBC, the tool is called the `ODBC Data Source Administrator`). If your operating system doesn't include an ODBC administrator, third-party options are available online.

Sections `Adding a Data Source Definition in Windows` and `Adding a Data Source Definition in Linux` walk you through adding a data source in Windows and Linux using the graphical tools available for each operating system. During the process of defining a data source, you'll be asked to specify a set of connection properties. Section `EDB-ODBC Connection Properties` contains information about `optional` data source connection properties; you can specify connection properties with graphical tools or edit the `odbc.ini` file with a text editor.

5 EDB-ODBC Connection Properties

The following table describes the connection properties that you can specify through

the dialogs in the graphical connection manager tools, or in the `odbc.ini` file that defines a named data source. The columns identify the connection property (as it appears in the ODBC Administrator dialogs), the corresponding keyword (as it appears in the `odbc.ini` file), the default value of the property, and a description of the connection property.

Property	Keyword name	Default value	Description
Database	Database	None	The name of the database to which you are connecting.
Driver	Driver	EDB-ODBC	The name of the ODBC driver.
Server	Servername	Localhost	The name or IP address of the server that you are connecting to.
dbms_name Description User Name Password	dbms_name Description Username Password	EnterpriseDB	Database system. Either EnterpriseDB or PostgreSQL. Descriptive name of the data source. The name of the user that this data source uses to connect to the server. The password of the user associated with this named data source.
CPTimeout	CPTimeout	0	Number of seconds before a connection times out (in a connection pooling environment).
Port	Port	5444	The TCP port that the postmaster is listening on.
Protocol	Protocol	7.4	If specified, forces the driver to use the given protocol version.

Property	Keyword name	Default value	Description
Level of Rollback on Errors	Use the Protocol option to specify rollback behavior.	Transaction Level	Specifies how the driver handles errors: 0 - Don't rollback 1 - Rollback the transaction 2 - Rollback the statement
Usage Count	UsageCount	1	The number of installations using this driver.
Read Only	ReadOnly	No	Specifies that the connection is READONLY.
Show System Tables	ShowSystemTables	No	If enabled, the driver reports system tables in the result set of the SQLTables() function.
OID Options: Show Column	ShowOidColumn	No	If enabled, the SQLColumns() function reports the OID column.
OID Options: Fake Index	FakeOidIndex	No	If enabled, the SQLStatistics() function reports that a unique index exists on each OID column.
Keyset Query Optimization	Ksqo	On	If enabled, enforces server-side support for keyset queries (generated by the MS Jet database engine).
Recognize Unique Indexes	UniqueIndex	On	If enabled, the SQLStatistics() function will report unique indexes. If not enabled, the SQLStatistics() function reports that indexes allow duplicate values.
Use Declare/Fetch	UseDeclareFetch	Off	If enabled, the driver will use server-side cursors. To enable UseDeclareFetch, specify a value of 1; to disable UseDeclareFetch, specify a value of 0.

Property	Keyword name	Default value	Description
CommLog	CommLog	Off	If enabled, records all client/server traffic in a log file.
Parse Statements	Parse	Off	If enabled, the driver parses simple SELECT statements when you call the SQLNumResultCols(), SQLDescribeCol() or SQLColAttributes() functions.
Cancel as FreeStmt	CancelAsFreeStmt	Off	If enabled, the SQLCancel() function will call SQLFreeStmt(SQL_Close) on your behalf.
MyLog	Debug	Off	If enabled, the driver records its work in a log file. On Windows, the file name is C:mylog<process-id>; and on Linux the file name is /tmp/mylog<username><process-id>.log.
Unknown Sizes	UnknownSizes	Maximum	Determines how the SQLDescribeCol() and SQLColAttributes() functions compute the size of a column. Specify 0 to force the driver to report the maximum size allowed for the type; specify 1 to force the driver to report an unknown length or 2 to force the driver to search the result set to find the longest value. Do not specify 2 if you have enabled UseDeclareFetch.

Property	Keyword name	Default value	Description
Text as LongVarchar	TextAsLongVarChar	8190	If enabled, the driver treats TEXT columns as if they are of type SQL_LONGVARCHAR. If disabled, the driver treats TEXT columns as SQL_VARCHAR values.
Unknown as Long Varchar	LongVarChar	False	If enabled, the driver treats values of unknown type as SQL_LONGVARCHAR values. If unchecked, the driver will treat values of unknown type as SQL_VARCHAR values. By default, values of unknown type are treated as Y values.
Bools as Char	BoolsAsChar	On	If enabled, the driver treats BOOL columns as SQL_CHAR values. If disabled, BOOL columns are treated as SQL_BIT values.
Max Varchar	MaxVarcharSize	255	If enabled, the driver treats VARCHAR and BPCHAR values longer than MaxVarCharSize as SQL_LONGVARCHAR values

Property	Keyword name	Default value	Description
Max Long Varchar Size	MaxLongVarcharSize	8190	If TextAsLongVarChar is on, the driver reports TEXT values are MaxLongVarcharSize bytes long. If UnknownAsLongVarChar is on, columns of unknown type are MaxLongVarcharSize bytes long; otherwise, they are reported to be MaxVarcharSize bytes in length.
Cache Size	Fetch	100	Determines the number of rows fetched by the driver when UseDeclareFetch is enabled.
SysTable Prefixes	ExtraSysTablePrefixes	dd;	Use the SysTablePrefixes field to specify a semi-colon delimited list of prefixes that indicate that a table is a system table. By default, the list contains dd;.
Cumulative Row Count for Insert	MapSqlParcNoBatch	Off/0	If enabled, the SQLRowCount() function will return a single, cumulative row count for the entire array of parameter settings for an INSERT statement. If disabled, an individual row count will be returned for each parameter setting. By default, this option is disabled.

Property	Keyword name	Default value	Description
LF<-> CR/LF conversion	LFConversion	System Dependent	The LF<->CR/LF conversion option instructs the driver to convert line-feed characters to carriage-return/line-feed pairs when fetching character values from the server and convert carriage-return/line-feed pairs back to line-feed characters when sending character values to the server. By default, this option is enabled.
Updatable Cursors	UpdatableCursors	Off	Permits positioned UPDATE and DELETE operations using the SQLSetPos() or SQLBulkOperations() functions.
Bytea as Long VarBinary	ByteaAsLongVarBinary	Off	If enabled, the driver treats BYTEA values as if they are of type SQL_LONGVARBINARY. If disabled, BYTEA values are treated as SQL_VARBINARY values.
Bytea as LO	ByteaAsLO	False	If enabled, the driver treats BYTEA values as if they are large objects.
Row versioning	RowVersioning	Off	The Row Versioning option specifies if the driver should include the xmin column when reporting the columns in a table. The xmin value is the ID of the transaction that created the row. You must use row versioning if you plan to create cursors where SQL_CONCURRENCY = SQL_CONCUR_ROWVER.

Property	Keyword name	Default value	Description
Disallow Premature	DisallowPremature	No/0	Determines driver behavior if you try to retrieve information about a query without executing the query. If Yes, the driver declares a cursor for the query and fetches the meta-data from the cursor. If No, the driver executes the command as soon as you request any meta-data.
True is -1	TrueIsMinus1	Off/0	TrueIsMinus1 tells the driver to return BOOL values of TRUE as -1. If this option is not enabled, the driver will return BOOL values of TRUE as 1. The driver always returns BOOL values of FALSE as 0.
Server side prepare	UseServerSidePrepare	No/0	If enabled, the driver uses the PREPARE and EXECUTE commands to implement the Prepare/Execute model.
Use GSSAPI for GSS request	GssAuthUseGSS	False/0	If set to True/1, the driver will send a GSSAPI authentication request to the server. Windows only.

Property	Keyword name	Default value	Description
Int8 As	BI	0	The value of BI determines how the driver treats BIGINT values: If -5 as a SQL_BIGINT, If 2 as a SQL_NUMERIC, If 8 as a SQL_DOUBLE, If 4 as a SQL_INTEGER, If 12 as a SQL_VARCHAR, If 0 (on an MS Jet client), as a SQL_NUMERIC, If 0 on any other client, as a SQL_BIGINT.

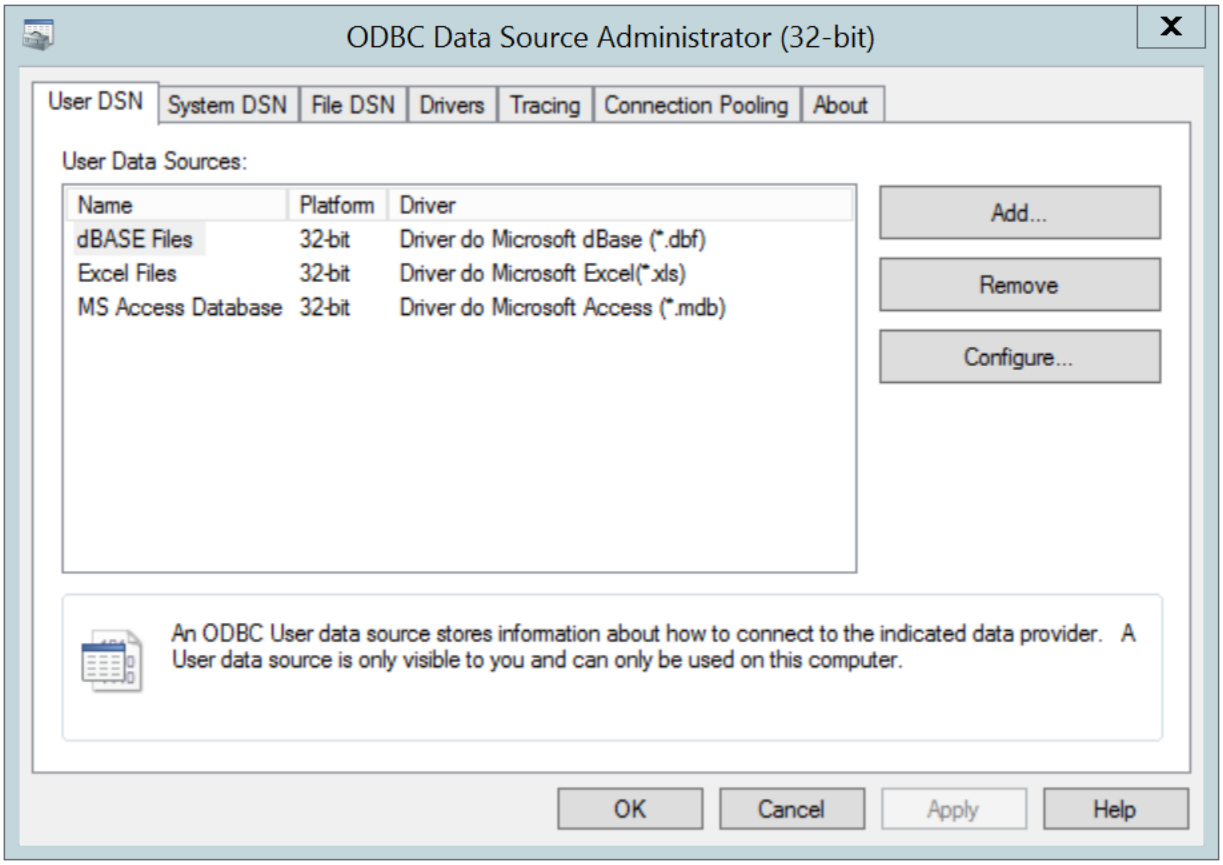
Property	Keyword name	Default value	Description
Extra options Connect Settings	AB ConnSettings	0x0	0x1 - Forces the output of short-length formatted connection strings. Specify this option if you are using the MFC CDatabase class. 0x2 - Allows MS Access to recognize PostgreSQL's serial type as AutoNumber type. 0x4 - Return ANSI character types for the inquiries from applications. Specify this option for applications that have difficulty handling Unicode data. 0x8 - If set, NULL dates are reported as empty strings and empty strings are interpreted as NULL dates on input. 0x10 - Determines if SQLGetInfo returns information about all tables, or only accessible tables. If set, only information is returned for accessible tables. 0x20 - If set, each SQL command is processed in a separate network round-trip, otherwise, SQL commands are grouped into as few round-trips as possible to reduce network latency. Contains a semicolon-delimited list of SQL commands that are executed when the driver

Property	Keyword name	Default value	Description
			connects to the server.
	Socket	4096	Specifies the buffer size that the driver uses to connect to the client.
	Lie	Off	If enabled, the driver claims to support unsupported ODBC features.
Lowercase Identifier	LowerCaseIdentifier	Off	If enabled, the driver translates identifiers to lowercase.
Disable Genetic Optimizer	Optimizer	Yes/1	Disables the genetic query optimizer.
Allow Keyset	UpdatableCursors	Yes/1	Allow Keyset driven cursors
SSL mode	SSLMode	Disabled	If libpq (and its dependencies) are installed in the same directory as the EDB-ODBC driver, enabling SSL Mode allows you to use SSL and other utilities.
Force Abbreviated Connection String	CX	No/0	Enables the option to force abbreviation of connection string.
Fake MSS	FakeOidIndex	No/0	Impersonates MS SQL Server enabling MS Access to recognize PostgreSQL's serial type as AutoNumber type.
BDE Environment	BDE	No/0	Enabling this option tunes EDB-ODBC to cater to Borland Database Engine compliant output (related to Unicode).
XA_Opt	INI_XAOPT	Yes/1	If enabled, calls to SQL_TABLES only include user-accessible tables.

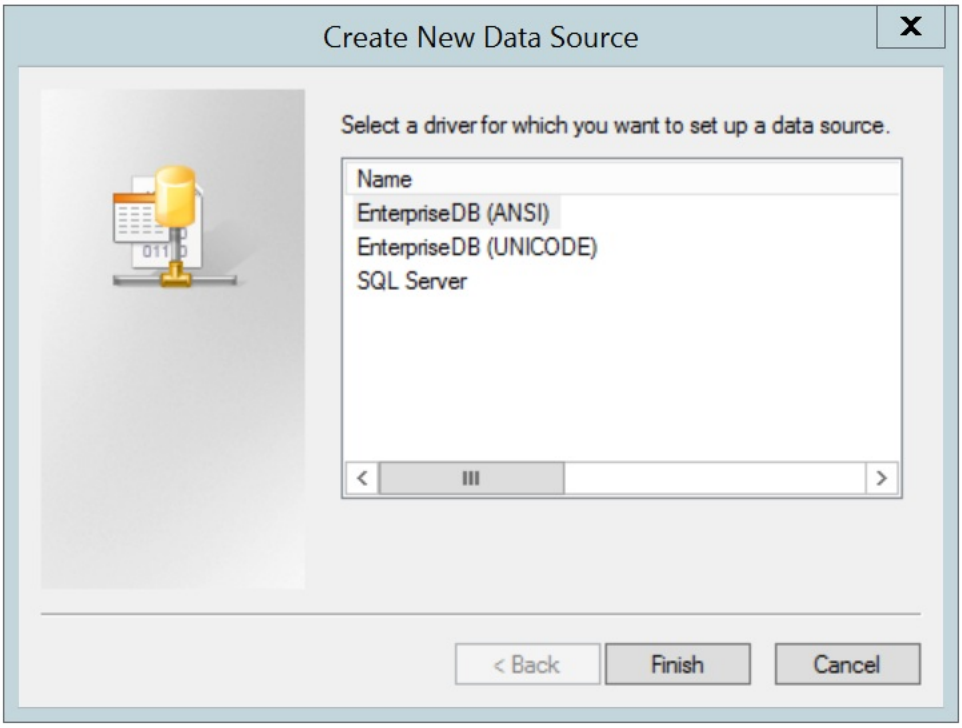
Adding a Data Source Definition in Windows

Adding data source definition in windows

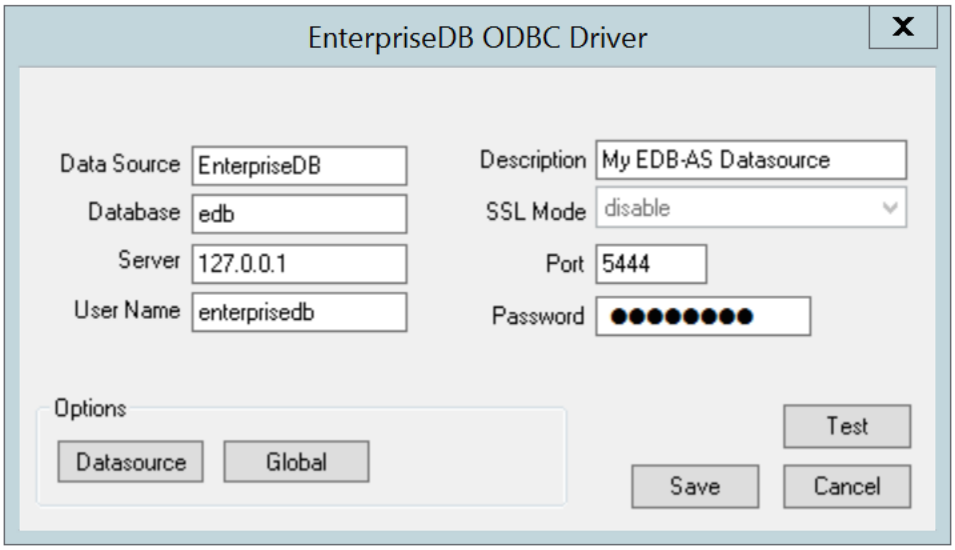
The Windows ODBC Data Source Administrator is a graphical interface that creates named data sources. You can open the ODBC Data Source Administrator by navigating to the Control Panel, opening the Administrative Tools menu, and double-clicking the appropriate ODBC Data Sources icon (32- or 64- bit).



Click the Add button to open the Create New Data Source dialog. Choose EnterpriseDB (ANSI) or EnterpriseDB (UNICODE) from the list of drivers and click Finish.



The EnterpriseDB ODBC Driver dialog opens.

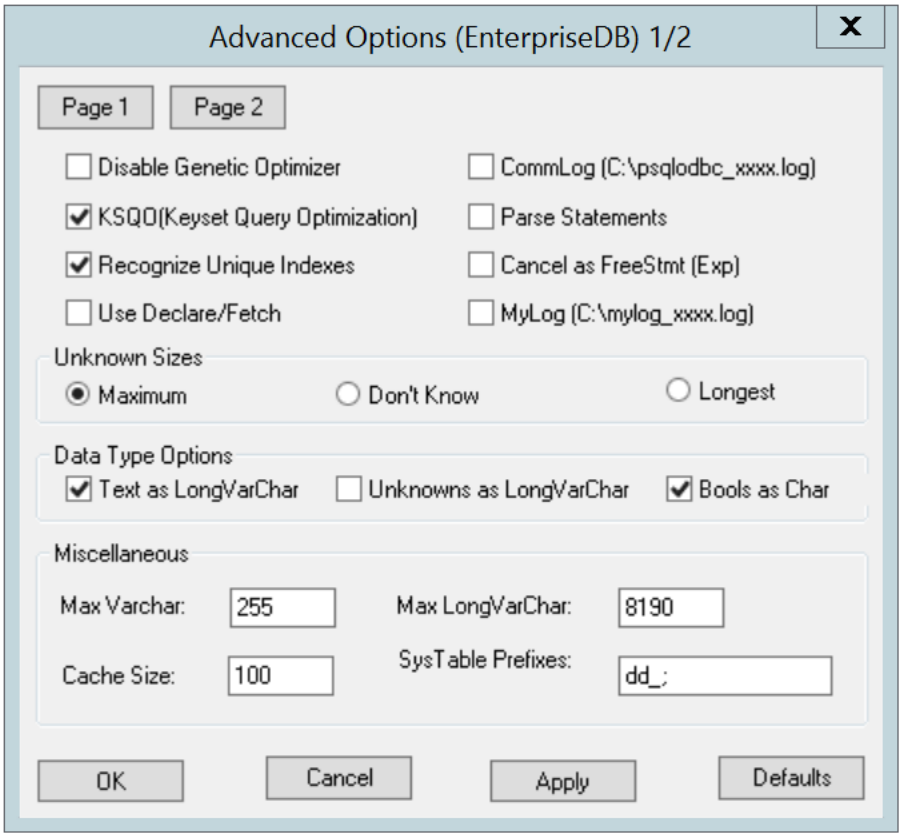


Use the fields on the dialog to define the named data source:

- Enter the Database name in the Database field.
- Enter the host name or IP address of Advanced Server in the Server field.
- Enter the name of a user in the User Name field.
- Enter a descriptive name for the named data source in the Description field.
- If libpq is installed in the same directory as the EDB-ODBC driver, the drop-down listbox next to the SSL Mode label will be active, allowing you to use SSL and other Advanced Server utilities.
- Accept the default port number (5444), or enter an alternative number in the Port field.
- Enter the password of the user in the Password field.

Use the Datasource button (located in the Options box) to open the Advanced Options dialog and specify connection properties.

The Global button opens a dialog on which you can specify logging options for the EDB-ODBC driver (not the data source, but the driver itself).



- Check the box next to Disable Genetic Optimizer to disable the genetic query optimizer. By default, the query optimizer is on.
- Check the box next to KSQO (Keyset Query Optimization) to enable server-side support for keyset queries. By default, Keyset Query Optimization is on.
- Check the box next to Recognize Unique Indexes to force the SQLStatistics() function to report unique indexes; if the option is not checked, the SQLStatistics() function will report that all indexes allow duplicate values. By default, Recognize Unique Indexes is on.
- Check the box next to Use Declare/Fetch to specify that the driver should use server-side cursors whenever your application executes a SELECT command. By default, Use Declare/Fetch is off.
- Check the box next to CommLog (C:\psqlodbc_xxxx.log) to record all client/server traffic in a log file. By default, logging is off.
- Check the box next to Parse Statements to specify that the driver (rather than the server) should attempt to parse simple SELECT statements when you call the SQLNumResultCols(), SQLDescribeCol(), or SQLColAttributes() function. By default, this option is off.
- Check the box next to Cancel as FreeStmt (Exp) to specify that the SQLCancel() function should call SQLFreeStmt(SQLClose) on your behalf. By default, this option is off.

- Check the box next to MyLog (C:\mylog_xxxx.log) to record a detailed record of driver activity in a log file. The log file is named c:\mylog_\ *process-id*.log. By default, logging is off.

The radio buttons in the Unknown Sizes box specify how the SQLDescribeCol() and SQLColAttributes() functions compute the size of a column of unknown type (see Section Supported Data Types for a list of known data types).

- Choose the button next to Maximum to specify that the driver report the maximum size allowed for a VARCHAR or LONGVARCHAR (dependent on the Unknowns as LongVarChar setting). If Unknowns as LongVarChar is enabled, the driver returns the maximum size of a LONGVARCHAR (specified in the Max LongVarChar field in the Miscellaneous box). If Unknowns as LongVarChar is not enabled, the driver returns the size specified in the Max VarChar field (in the Miscellaneous box).
- Choose the button next to Don't know to specify that the driver report a length of "unknown".
- Choose the button next to Longest to specify that the driver search the result set and report the longest value found. (Note: you should not specify Longest if UseDeclareFetch is enabled.)

The properties in the Data Type Options box determine how the driver treats columns of specific types:

- Check the box next to Text as LongVarChar to treat TEXT values as if they are of type SQL_LONGVARCHAR. If the box is not checked, the driver will treat TEXT values as SQL_VARCHAR values. By default, TEXT values are treated as SQL_LONGVARCHAR values.
- Check the box next to Unknowns as LongVarChar to specify that the driver treat values of unknown type as SQL_LONGVARCHAR values. If unchecked, the driver will treat values of unknown type as SQL_VARCHAR values. By default, values of unknown type are treated as SQL_VARCHAR values.
- Check the box next to Bools as Char to specify that the driver treat BOOL values as SQL_CHAR values. If unchecked, BOOL values are treated as SQL_BIT values. By default, BOOL values are treated as SQL_CHAR values.

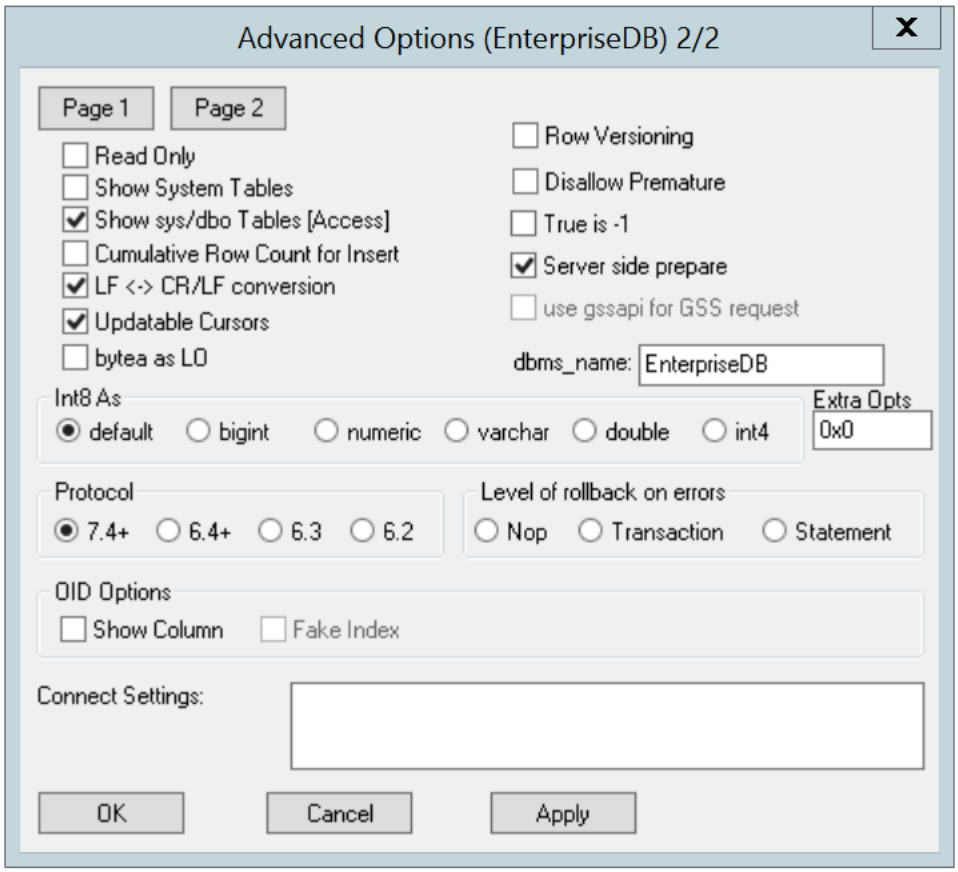
You can specify values for some of the properties associated with the named data source in the fields in the Miscellaneous box:

- Indicate the maximum length allowed for a VARCHAR value in the Max VarChar field. By default, this value is set to 255.
- Enter the maximum length allowed for a LONGVARCHAR value in the Max LongVarChar field. By default, this value is set to 8190.
- Specify the number of rows fetched by the driver (when UseDeclareFetch is enabled) in the Cache Size field. The default value is 100.
- Use the SysTablePrefixes field to specify a semi-colon delimited list of prefixes that indicate that a table is a system table. By default, the list contains dd_;.

You can reset the values on this dialog to their default settings by choosing the Defaults button.

Click the Apply button to apply any changes to the data source properties, or the Cancel button to exit the dialog without applying any changes. Choose the OK button to apply any changes to the dialog and exit.

Select the Page 2 button (in the upper-left hand corner of the Advanced Options dialog) to access a second set of advanced options.



- Check the box next to Read Only to prevent the driver from executing the following commands: INSERT, UPDATE, DELETE, CREATE, ALTER, DROP, GRANT, REVOKE or LOCK. Invoking the Read Only option also prevents any calls that use ODBC's procedure call escape syntax (call=procedure-name?). By default, this option is off.
- Check the box next to Show System Tables to include system tables in the result set of the SQLTables() function. If the option is enabled, the driver will include any table whose name starts with pg_ or any of the prefixes listed in the SysTablePrefixes field of Page 1 of the Advanced Options dialog. By default, this option is off.
- Check the box next to Show sys/dbo Tables [Access] to access objects in the sys schema and dbo schema through the ODBC data source. By default, this option is enabled (checked).
- Check the box next to Cumulative Row Count for Insert to cause a single, cumulative row count to be returned for the entire array of parameter settings for

an INSERT statement when a call to the SQLRowCount() method is performed. If this option is not enabled (the box is not checked), then an individual row count is available for each parameter setting in the array, and thus, a call to SQLRowCount() returns the count for the last inserted row.

- Check the box next to LF<->CR/LF conversion to instruct the driver to convert line-feed characters to carriage-return/line-feed pairs when fetching character values from the server and convert carriage-return/line-feed pairs back to line-feed characters when sending character values to the server. By default, this option is enabled.
- Check the box next to Updatable Cursors to specify that the driver should permit positioned UPDATE and DELETE operations with the SQLSetPos() or SQLBulkOperations() functions. By default, this option is enabled.
- Check the box next to bytea as LO to specify that the driver should treat BYTEA values as if they are SQL_LONGVARBINARY values. If the box is not checked, EDB-ODBC will treat BYTEA values as if they are SQL_VARBINARY values. By default, BYTEA values are treated as SQL_VARBINARY values.
- Check the box next to Row Versioning to include the xmin column when reporting the columns in a table. The xmin column is the ID of the transaction that created the row. You must use row versioning if you plan to create cursors where SQL_CONCURRENCY = SQL_CONCUR_ROWVER. By default, Row Versioning is off.
- Check the box next to Disallow Premature to specify that the driver should retrieve meta-data about a query (i.e., the number of columns in a result set, or the column types) without actually executing the query. If this option is not specified, the driver executes the query when you request meta-data about the query. By default, Disallow Premature is off.
- Check the box next to True is -1 to tell the driver to return BOOL values of True as a -1. If this option is not enabled, the driver will return BOOL values of True as 1. The driver always returns BOOL values of False as 0.
- Check the box next to Server side prepare to tell the driver to use the PREPARE and EXECUTE commands to implement the Prepare/Execute model. By default, this box is checked.
- Check the box next to use gssapi for GSS request to instruct the driver to send a GSSAPI connection request to the server.
- Enter the database system (either EnterpriseDB or PostgreSQL) in the dbms_name field. The value entered here is returned in the SQL_DBMS_NAME argument when the SQLGetInfo() function is called. The default is EnterpriseDB.

Use the radio buttons in the Int8 As box to specify how the driver should return BIGINT values to the client. Select the radio button next to default to specify the default type of NUMERIC if the client is MS Jet, BIGINT if the client is any other ODBC client. You can optionally specify that the driver return BIGINT values as a bigint (SQL_BIGINT), numeric (SQL_NUMERIC), varchar (SQL_VARCHAR), double (SQL_DOUBLE), or int4 (SQL_INTEGER).

The default value of the Extra Opts field is 0x0. Extra Opts may be:

Option	Specifies
0x1	Forces the output of short-length formatted connection string. Select this option when you are using the MFC CDatabase class.
0x2	Allows MS Access to recognize PostgreSQL's serial type as AutoNumber type.
0x4	Return ANSI character types for the inquiries from applications. Select this option for applications that have difficulty handling Unicode data.
0x8	If set, NULL dates are reported as empty strings and empty strings are interpreted as NULL dates on input.
0x10	Determines if SQLGetInfo returns information about all tables, or only accessible tables. If set, only information is returned for accessible tables.
0x20	If set, each SQL command is processed in a separate network round-trip, otherwise, SQL commands are grouped into as few round-trips as possible to reduce network latency.

The Protocol box contains radio buttons that tell the driver to interact with the server using a specific front-end/back-end protocol version. By default, the Protocol selected is 7.4+; you can optionally select from versions 6.4+, 6.3 or 6.2.

The Level of Rollback on errors box contains radio buttons that specify how the driver handles error handling:

Option	Specifies
Transaction	If the driver encounters an error, it will rollback the current transaction.
Statement	If the driver encounters an error, it will rollback the current statement.
Nop	If the driver encounters an error, you must manually rollback the current transaction before the application can continue.

The OID Options box contains options that control the way the driver exposes the OID column contained in some tables:

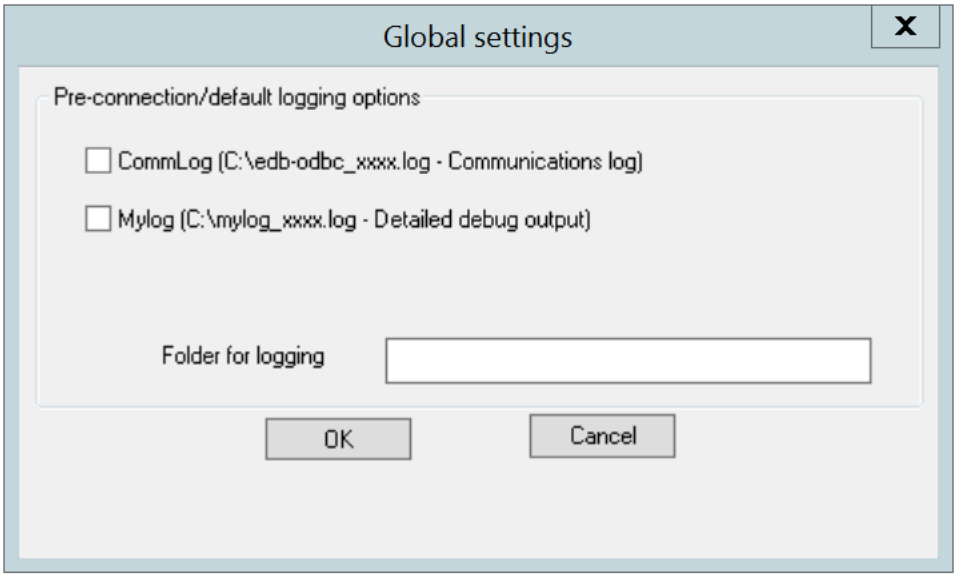
- Check the box next to Show Column to include the OID column in the result set of the SQLColumns() function. If this box is not checked, the OID column is hidden from SQLColumns().
- Check the box next to Fake Columns to specify that the SQLStatistics() function should report that a unique index exists on each OID column.

Use the Connect Settings field to specify a list of parameter assignments that the driver will use when opening this connection. Any configuration parameter that you can modify with a SET statement can be included in the semi-colon delimited list. For example:

```
set search_path to company1,public;
```

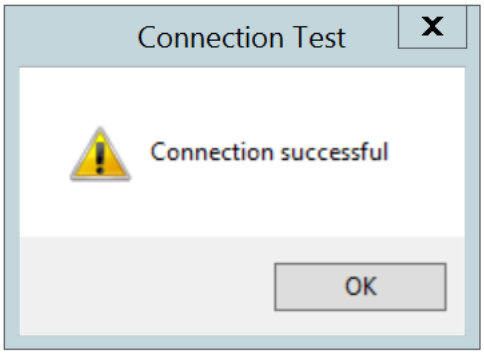
When you've defined the connection properties for the named data source, click the Apply button to apply the options; you can optionally exit without saving any options by choosing Cancel. Select the OK button to save the options and exit.

Choose the Global button (on the EnterpriseDB ODBC Driver dialog) to open the Global Settings dialog. The options on this dialog control logging options for the EDB-ODBC driver. Use this dialog to enforce logging when the driver is used without a named data source, or for logging driver operations that occur before the connection string is parsed.



- Check the box next to the CommLog field to record all client/server traffic in a log file. The logfile is named C:\psqlodbc_process-id where process-id is the name of the process in use.
- Check the box next to the Mylog field to keep a logfile of the driver's activity. The logfile is named c:\mylog_process-id where process-id is the name of the process in use.
- Specify a location for the logfiles in the Folder for logging field.

When you've entered the connection information for the named data source, click the Test button to verify that the driver manager can connect to the defined data source.



Click the OK button to exit Connection Test dialog. If the connection is successful, click the Save button to save the named data source. If there are problems establishing a connection, adjust the parameters and test again.

Adding a Data Source Definition in Linux

Adding data source definition in linux

The Linux ODBC Administrator is a graphical tool that is distributed with unixODBC; you can use the ODBC Administrator to manage ODBC drivers and named resources. To add the ODBC Administrator to your system, open a terminal window, assume superuser privileges, and enter:

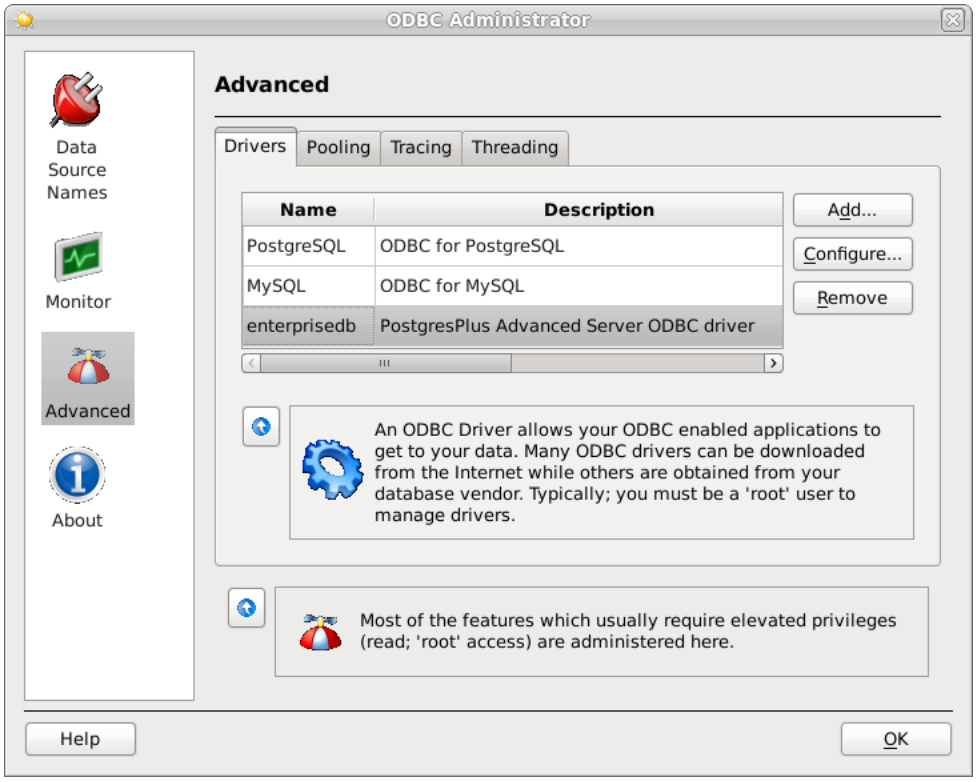
```
yum install unixODBC
```

followed by:

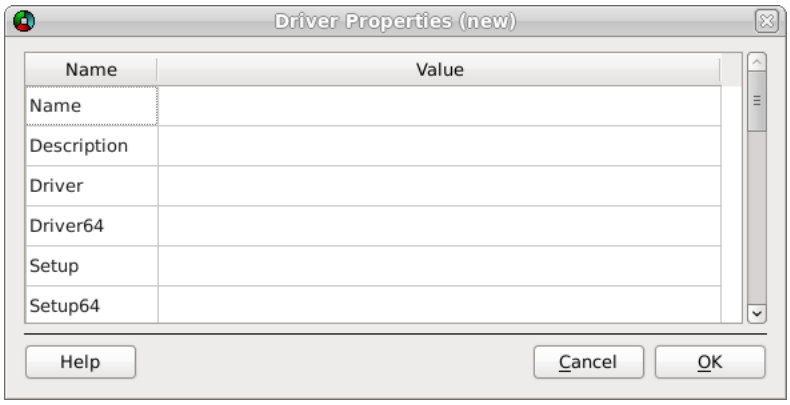
```
yum install unixODBC-kde
```

To invoke the ODBC Administrator, open a terminal window and enter ODBCConfig.

When you install the Advanced Server Connectors component, the EDB-ODBC driver is added to the list of drivers in the ODBC Administrator. Click Advanced, and then select the Drivers tab to verify that the enterprisedb driver appears in the list.



If the EDB-ODBC driver does not appear in the list of drivers, you can add it using the ODBC Administrator. To add a driver definition, select the Drivers tab, and click Add. The Driver Properties (new) window opens, as shown below:



Complete the Driver Properties window to register the EDB-ODBC driver with the driver manager:

- Add a unique name for the driver to the Name field.
- Add a driver description to the Description field.
- Add the path to the location of the EDB-ODBC driver in the Driver field. By default, the complete path to the driver is:

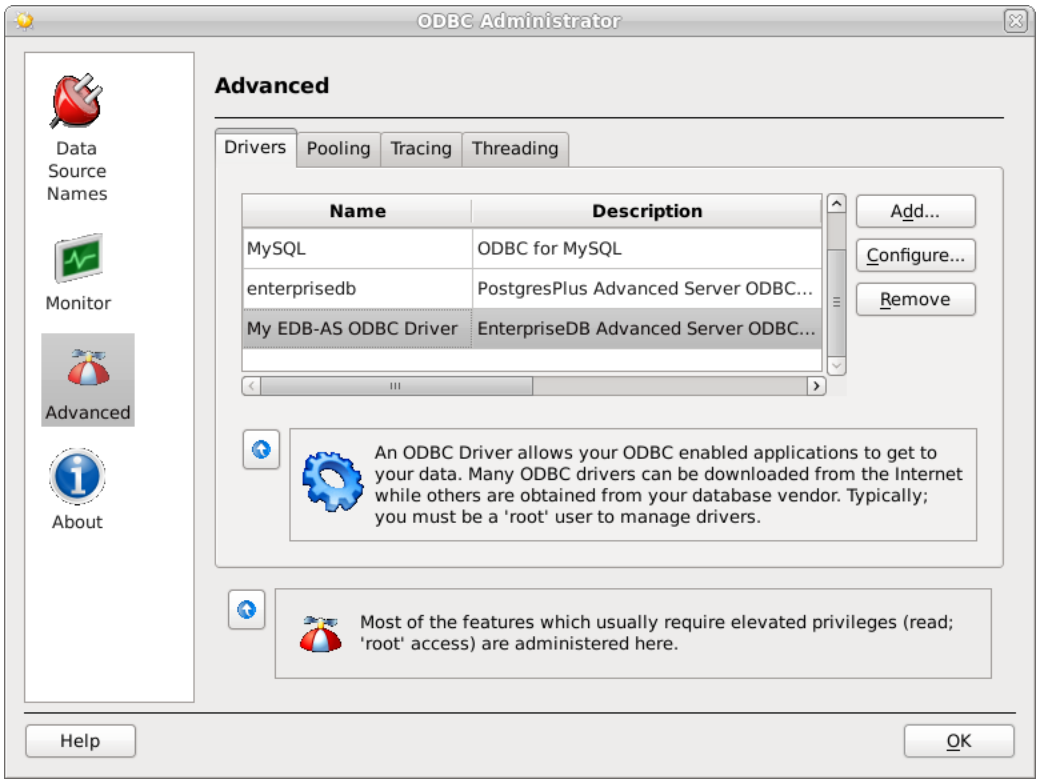
  /usr/edb/odbc/lib/edb-odbc.so

- Add the path to the location of the EDB-ODBC driver setup file in the Setup field.

By default, the complete path to the driver setup file is:

```
/usr/edb/odbc/lib/libodbcedbS.so
```

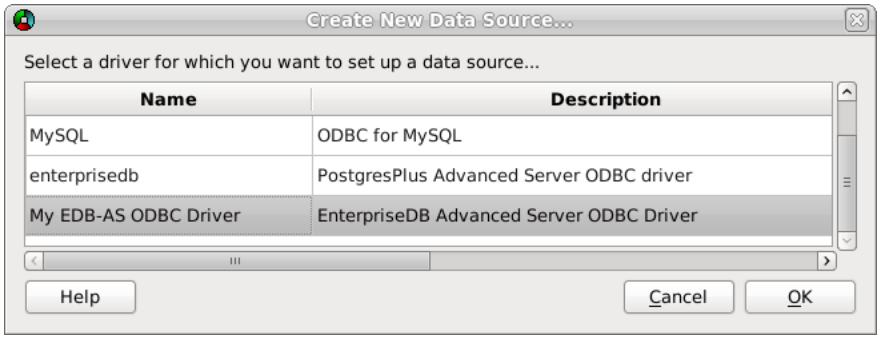
When you've described the driver properties for the EDB-ODBC driver, click OK. The ODBC Data Source Administrator window now includes the EDB-ODBC driver in the list of available ODBC drivers.



With the EDB-ODBC driver available to the driver manager, you can add a data source. Click the Data Source Names option in the left panel, and then choose the appropriate DSN tab for the type of data source name you would like to add:

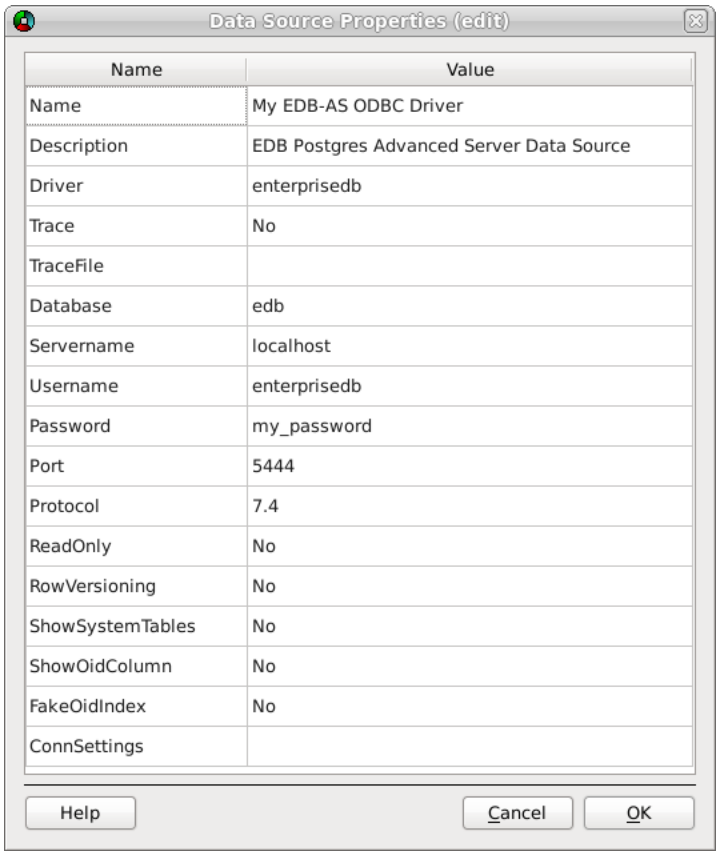
- Choose the User tab to add a named data source that is available only to the current user (the data source will be stored in /user/.odbc.ini).
- Choose the System tab add a named data source that is available to all users. All system data sources are stored in a single file (usually /etc/odbc.ini).
- Choose the File tab to add a named data source that is available to all users, but that is stored in a file of your choosing.

Select the appropriate tab and click Add. The Create a New Data Source… window opens, as shown below:



Select the EDB-ODBC driver from the list, and click OK to open the Data Source Properties window.

Complete the Data Source Properties (new) window, specifying the connection properties for the EDB-ODBC driver.



- Enter the data source name in the Name field.
- Enter a description of the named data source in the Description field.
- The unixODBC driver includes a trace utility that records the sequence of calls made an ODBC application to a log file. Specify Yes in the Trace field to turn the trace utility on. Note that using the trace utility can slow down an application.
- Use the TraceFile field to specify a file to receive information returned by the Trace utility.
- Enter the name of the Advanced Server database in the Database field.
- Enter the host name or IP address of Advanced Server in the Servername field.
- Enter the name of a user in the Username field.
- Enter the password for the user in the Password field.

- Enter a port number (or accept the default value of 5444) in the Port field.
- Use the Protocol field to specify a front-end/back-end protocol version; the default value is 7.4. You can optionally select from protocol versions 7.4, 6.4, 6.3 or 6.2.
- Use the ReadOnly field to specify Yes to prevent the driver from executing the following commands: INSERT, UPDATE, DELETE, CREATE, ALTER, DROP, GRANT, REVOKE or LOCK. Enabling the Read Only option also prevents any calls that use the ODBC procedure call escape syntax (call=procedure-name?). By default, ReadOnly is set to No.
- Use the RowVersioning field to specify Yes if the driver should include the xmin column when reporting the columns in a table. The xmin column is the ID of the transaction that created the row. You must use row versioning if you plan to create cursors where SQL_CONCURRENCY = SQL_CONCUR_ROWVER. By default, Row Versioning is set to No.
- Use the ShowSystemTables field to specify Yes if the driver should include system tables in the result set of the SQLTables() function. By default, this field is set to No.
- Use the ShowOidColumn field to specify Yes if the driver should include the OID column in the result set of the SQLColumns() function. If ShowOidColumn is set to No, the OID column is hidden from SQLColumns(). By default, this option is set to No.
- Use the FakeOidIndex field to specify Yes if the SQLStatistics() function should report that a unique index exists on each OID column. This is useful when your application needs a unique identifier and your table doesn't include one. The default value is No.
- Use the ConnSettings field to specify a list of parameter assignments that the driver will use when opening this connection.

When you've defined the connection properties, click OK.

The new data source is added to the list of data source names: