



EDB

EDB ODBC Connector

Release 12.2.0.2

EDB

May 27, 2022

CONTENTS

1	Release notes	3
1.1	Version 13.02.0000.01	3
1.2	Version 13.01.0000.02	3
1.3	Version 13.01.0000.01	4
1.4	Version 13.00.0000.01	4
2	Product compatibility	5
3	Installing the EDB ODBC Connector	7
3.1	Installing on Linux	7
3.1.1	Installing EDB ODBC Connector on IBM Power (ppc64le)	8
3.1.2	Installing EDB ODBC Connector on Linux x86 (amd64)	11
3.2	Installing the ODBC Connector on Windows	19
4	Creating a data source	27
5	Connection properties	29
5.1	Adding a data source definition in Windows	34
5.2	Adding a data source definition in Linux	38
6	Driver functionality	41
6.1	SQLGetInfo()	41
6.2	Connection attributes	54
6.2.1	SQLGetConnectAttr()	54
6.2.2	SQLSetConnectAttr()	55
6.3	Environment attributes	56
6.3.1	SQLGetEnvAttr()	56
6.4	SQLSetEnvAttr()	57
6.5	Statement attributes	58
6.5.1	SQLGetStmtAttr()	58
6.5.2	SQLSetStmtAttr()	60
6.6	Error handling	60
6.6.1	SQLGetDiagRec()	60
6.7	Supported ODBC API functions	61
6.8	Supported data types	63
6.9	Prerequisite for ADO users	64
6.10	Thread safety	64
7	Scram compatibility	65
8	SLES 15 on IBM Power (ppc64le)	67

8.1	Setting up the Repository	67
8.2	Installing the Package	68
9	Upgrading a Linux installation	69

ODBC (Open Database Connectivity) is a programming interface that allows a client application to connect to any database that provides an ODBC driver. EDB ODBC Connector is an interface that allows an ODBC-compliant client application to connect to an EDB Postgres Advanced Server database. The EDB ODBC Connector allows an application that was designed to work with other databases to run on EDB Postgres Advanced Server. The ODBC Connector provides a way for the client application to establish a connection, send queries, and retrieve results from EDB Postgres Advanced Server.

While ODBC Connector provides a level of application portability, the portability is limited. It provides a connection but doesn't guarantee command compatibility. Commands that are acceptable in another database might not work in EDB Postgres 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, which handles named connections for Windows or Linux
- The database-specific ODBC driver (ODBC Connector)
- The ODBC-compliant server (EDB Postgres Advanced Server)

You can write client applications in any language that has a binding for ODBC. C, MS-Access, and C++ are just a few.

RELEASE NOTES

The ODBC JDBC connector documentation describes version 13 of the ODBC JDBC connector.

Release notes describe what's new in a release. When a minor or patch release introduces new functionality, indicators in the content identify the version that introduced the new feature.

Version	Release date
[13.02.0000.01](02_odbc_132000May17notes)	
[13.01.0000.02](03_odbc_132000Dec12notes)	
[13.01.0000.01](04_odbc_132000Sep14notes)	
[13.00.0000.01](05_odbc_132000Nov19notes)	

1.1 Version 13.02.0000.01

EDB ODBC Connector 13.02.0000.01 includes the following enhancement:

Type	Description
Upstream merge	Merged with the upstream community driver version 13.02.0000. See the community [Release notes](https://odbc.postgresql.org/docs/release.html) for details.
Bug fix	MS SQL linked server repetition of databases [Support ticket #1044249].

1.2 Version 13.01.0000.02

EDB ODBC Connector 13.01.0000.02 includes the following enhancement:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server version 14.1.0.

1.3 Version 13.01.0000.01

EDB ODBC Connector 13.01.0000.01 includes the following upstream merge:

Type	Description
Upstream merge	Merged with the upstream community driver version 13.01.0000. See the community [Release notes](https://odbc.postgresql.org/docs/release.html) for details.

1.4 Version 13.00.0000.01

EDB ODBC Connector 13.00.0000.01 includes the following upstream merge:

Type	Description
Upstream merge	Merged with the upstream community driver version 13.00.0000. See the community [Release notes](https://odbc.postgresql.org/docs/release.html) for details.

PRODUCT COMPATIBILITY

This table lists the latest ODBC Connector versions and their supported corresponding EDB Postgres Advanced Server (EPAS) versions. See

Product compatibility for details.

ODBC Connector	EPAS 14	EPAS 13	EPAS 12	EPAS 11	EPAS 10
[13.01.0.02](01_odbc_13.1.0.02_rel_notes)	Y	Y	Y	Y	Y
[13.01.0.01](03_odbc_13.1.0.01_rel_notes)	N	Y	Y	Y	Y
[13.00.0.01](05_odbc_13.0.0.01_rel_notes)	N	Y	Y	Y	Y

INSTALLING THE EDB ODBC CONNECTOR

You can install the EDB ODBC Connector on:

- *Installing on Linux*
- *Installing the ODBC Connector on Windows*

3.1 Installing on Linux

To install the EDB ODBC Connector, you must have credentials that allow access to the EnterpriseDB repository. To request credentials that allow you to access an EnterpriseDB repository, see the [EDB Repository Access instructions](#).

For platform-specific install instructions, see:

- Linux x86-64 (amd64):
 - *Installing EDB ODBC Connector on RHEL 8/OL 8 x86*
 - *Installing EDB ODBC Connector on Rocky Linux 8/AlmaLinux 8 x86*
 - *Installing EDB ODBC Connector on RHEL 7/OL 7 x86*
 - *Installing EDB ODBC Connector on CentOS 7 x86*
 - *Installing EDB ODBC Connector on SLES 15 x86*
 - *Installing EDB ODBC Connector on SLES 12 IBM Power (ppc64le)*
 - *Installing EDB ODBC Connector on Ubuntu 20.04/Debian 10 x86*
 - *Installing EDB ODBC Connector on Ubuntu 18.04/Debian 9 x86*
- Linux on IBM Power (ppc64le):
 - *Installing EDB ODBC Connector on RHEL 8 IBM Power (ppc64le)*
 - *Installing EDB ODBC Connector on RHEL 7 IBM Power (ppc64le)*
 - *Installing EDB ODBC Connector on SLES 12 IBM Power (ppc64le)*

3.1.1 Installing EDB ODBC Connector on IBM Power (ppc64le)

For operating system-specific install instructions, see:

- *Installing EDB ODBC Connector on RHEL 8 IBM Power (ppc64le)*
- *Installing EDB ODBC Connector on RHEL 7 IBM Power (ppc64le)*
- *Installing EDB ODBC Connector on SLES 12 IBM Power (ppc64le)*

Installing EDB ODBC Connector on RHEL 8 IBM Power (ppc64le)

There are two steps to completing an installation:

- Setting up the repository
- Installing the package

For each step, you must be logged in as superuser.

To log in as a superuser:

```
sudo su -
```

Setting up the repository

1. To register with EDB to receive credentials for the EDB repository, visit: [Repository Access Request](#).
2. Set up the EDB repository:

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

This creates the /etc/yum.repos.d/edb.repo configuration file.

1. Add your EDB credentials to the edb.repo file:

```
sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
```

Where USERNAME:PASSWORD is the username and password available from your [EDB account](#).

1. Install the EPEL repository and refresh the cache:

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

1. Enable the codeready-builder-for-rhel-8-*¹-rpms repository since EPEL packages may depend on packages from it:

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

1. Disable the built-in PostgreSQL module: edb_notranlate_5

Installing the package

```
dnf -y install edb-odbc
```

Installing EDB ODBC Connector on RHEL 7 IBM Power (ppc64le)

Before installing the ODBC Connector, you must install the following prerequisite packages, and request credentials from EDB:1. Use the following commands to install Advance Toolchain: `edb_notranlate_0`

2. 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 the ODBC Connector.

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
repo_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.

Installing EDB ODBC Connector on SLES 12 IBM Power (ppc64le)

There are two steps to completing an installation:

- Setting up the repository
- Installing the package

For each step, you must be logged in as superuser.

```
# To log in as a superuser:
sudo su -
```

Before setting up the repository, you need to register with EDB. To receive credentials for the EDB repository, visit: [Repository Access Request](#).

Setting up the repository

Setting up the repository is a one time task. If you have already set up your repository, you do not need to perform these steps.

```
# Install the repository configuration and enter your EDB repository
# credentials when prompted
zypper addrepo https://zypp.enterprisedb.com/suse/edb-sles.repo

# Install SUSEConnect to register the host with SUSE, allowing access to
# SUSE repositories
zypper install SUSEConnect

# Register the host with SUSE, allowing access to SUSE repositories
# Replace REGISTRATION_CODE and EMAIL with your SUSE registration
# information
SUSEConnect -r REGISTRATION_CODE -e EMAIL

# Activate the required SUSE modules
SUSEConnect -p PackageHub/12.5/ppc64le
SUSEConnect -p sle-sdk/12.5/ppc64le
```

(continues on next page)

(continued from previous page)

```
# Refresh the metadata
zypper refresh
```

Installing the package

```
zypper -n install edb-odbc
```

3.1.2 Installing EDB ODBC Connector on Linux x86 (amd64)

For operating system-specific install instructions, see:

- *Installing EDB ODBC Connector on RHEL 8/OL 8 x86*
- *Installing EDB ODBC Connector on Rocky Linux 8/AlmaLinux 8 x86*
- *Installing EDB ODBC Connector on RHEL 7/OL 7 x86*
- *Installing EDB ODBC Connector on CentOS 7 x86*
- *Installing EDB ODBC Connector on SLES 15 x86*
- *Installing EDB ODBC Connector on SLES 12 IBM Power (ppc64le)*
- *Installing EDB ODBC Connector on Ubuntu 20.04/Debian 10 x86*
- *Installing EDB ODBC Connector on Ubuntu 18.04/Debian 9 x86*

Installing EDB ODBC Connector on RHEL 8/OL 8 x86

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%202004-09-2019.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
repo_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.

Installing EDB ODBC Connector on Rocky Linux 8/AlmaLinux 8 x86

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
repo_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.

Installing EDB ODBC Connector on RHEL 7/OL 7 x86

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%202004-09-2019.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
repo_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.

Installing EDB ODBC Connector on CentOS 7 x86

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
repo_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.

Installing EDB ODBC Connector on SLES 15 x86

There are two steps to completing an installation:

- Setting up the repository
- Installing the package

For each step, you must be logged in as superuser.

```
# To log in as a superuser:
sudo su -
```

Before setting up the repository, you need to register with EDB. To receive credentials for the EDB repository, visit: [Repository Access Request](#).

Setting up the repository

Setting up the repository is a one time task. If you have already set up your repository, you do not need to perform these steps.

```
# Install the repository configuration and enter your EDB repository
# credentials when prompted
zypper addrepo https://zypp.enterprisedb.com/suse/edb-sles.repo

# Install SUSEConnect to register the host with SUSE, allowing access to
# SUSE repositories
zypper install SUSEConnect

# Register the host with SUSE, allowing access to SUSE repositories
# Replace REGISTRATION_CODE and EMAIL with your SUSE registration
# information
SUSEConnect -r REGISTRATION_CODE -e EMAIL

# Activate the required SUSE module
SUSEConnect -p PackageHub/15.3/x86_64

# Refresh the metadata
zypper refresh
```

Installing the package

```
zypper -n install edb-odbc
```

Installing EDB ODBC Connector on SLES 12 x86

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 EDB ODBC Connector on Ubuntu 20.04/Debian 10 x86

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:

3. 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
```

2. 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
```

Installing EDB ODBC Connector on Ubuntu 18.04/Debian 9 x86

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:

edb_notranlate_1 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
```

3.2 Installing the ODBC Connector on Windows

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).

The ODBC Connectors Installation wizard.

Click **Next** to continue.

The Installation dialog

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

The Ready to Install dialog.

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

The installation is complete.

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.

Starting StackBuilder Plus

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

Selecting the Connectors installer.

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

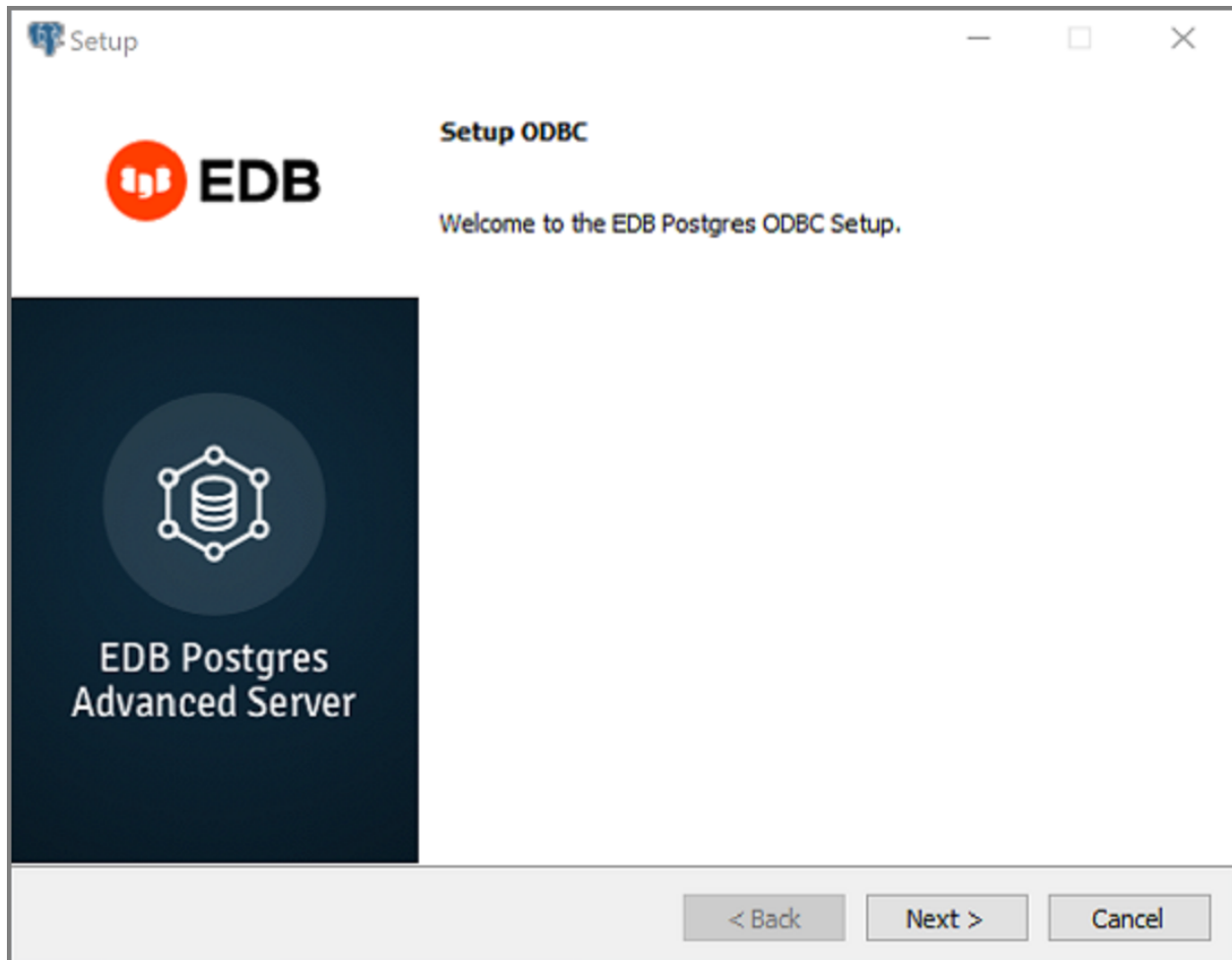


Fig. 1: The ODBC Connectors Installation wizard.

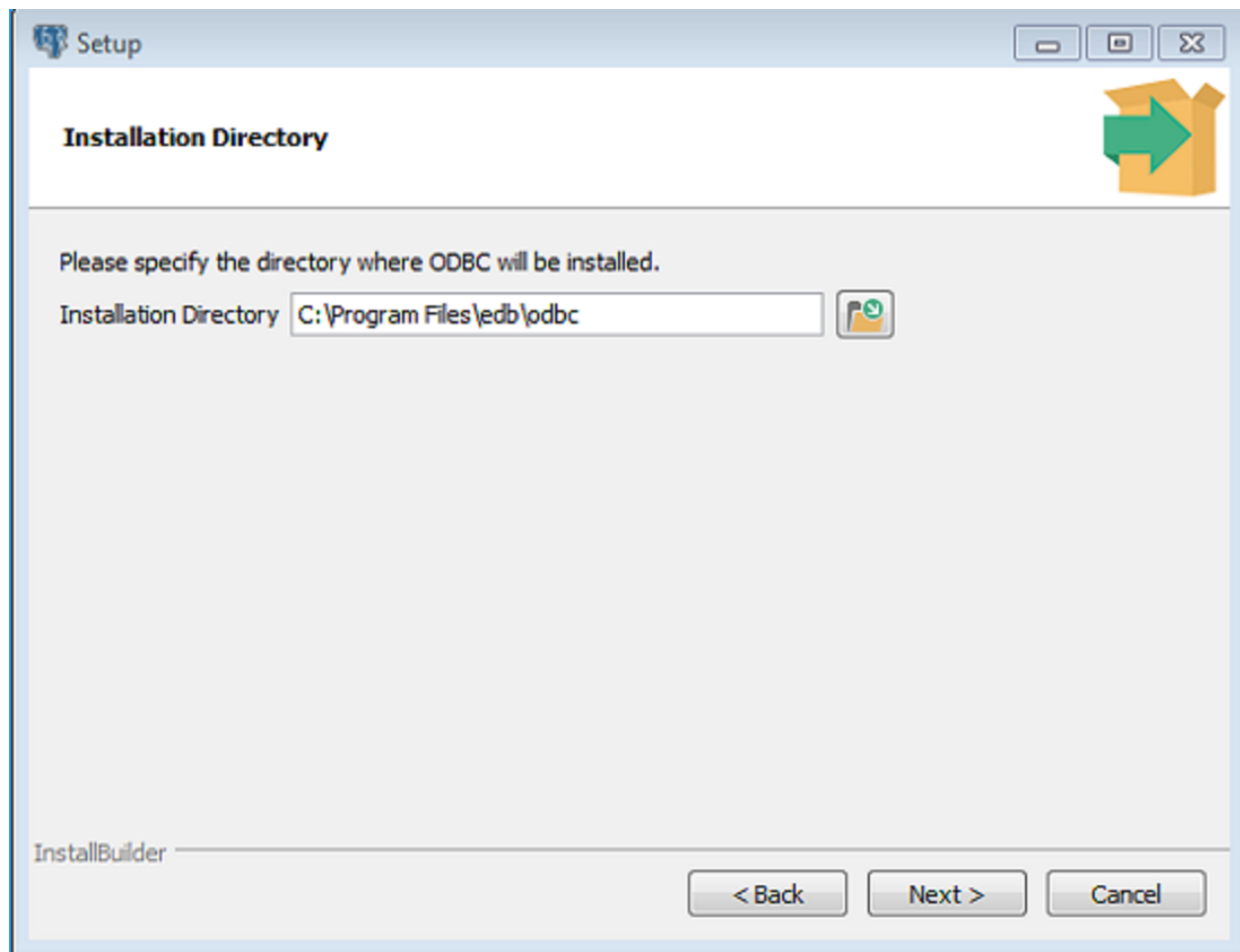


Fig. 2: The Installation dialog.

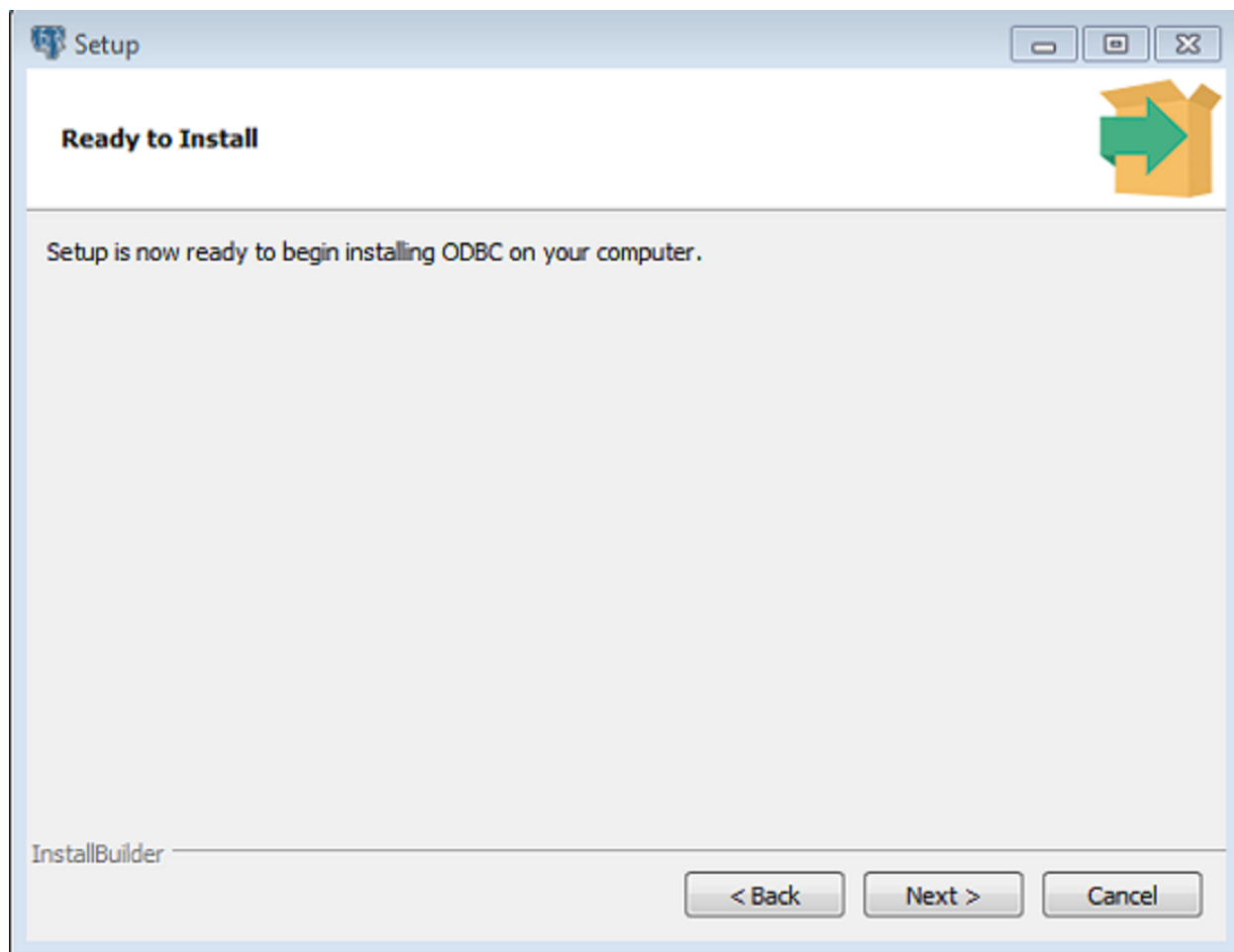


Fig. 3: The Ready to Install dialog.

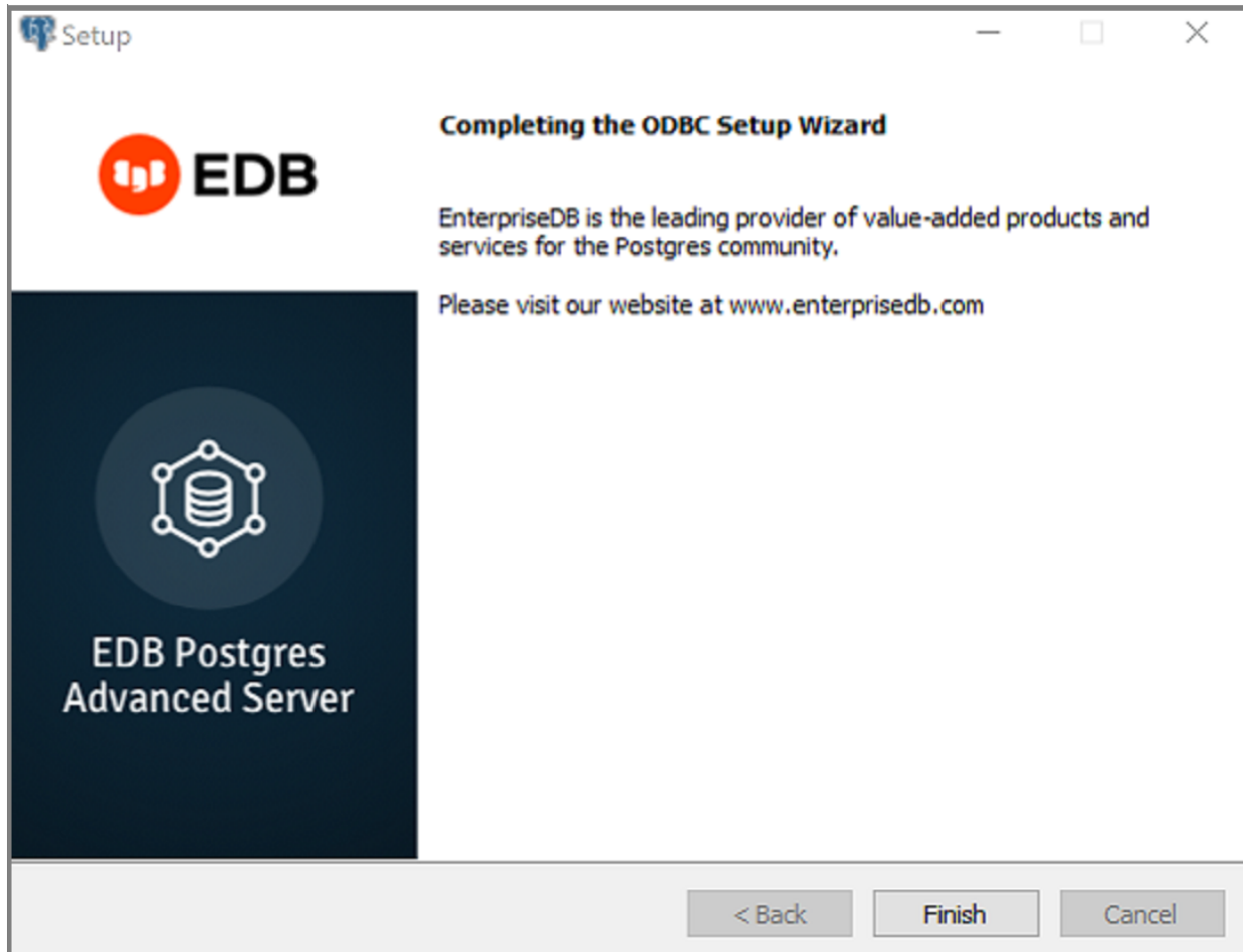


Fig. 4: The installation is complete.

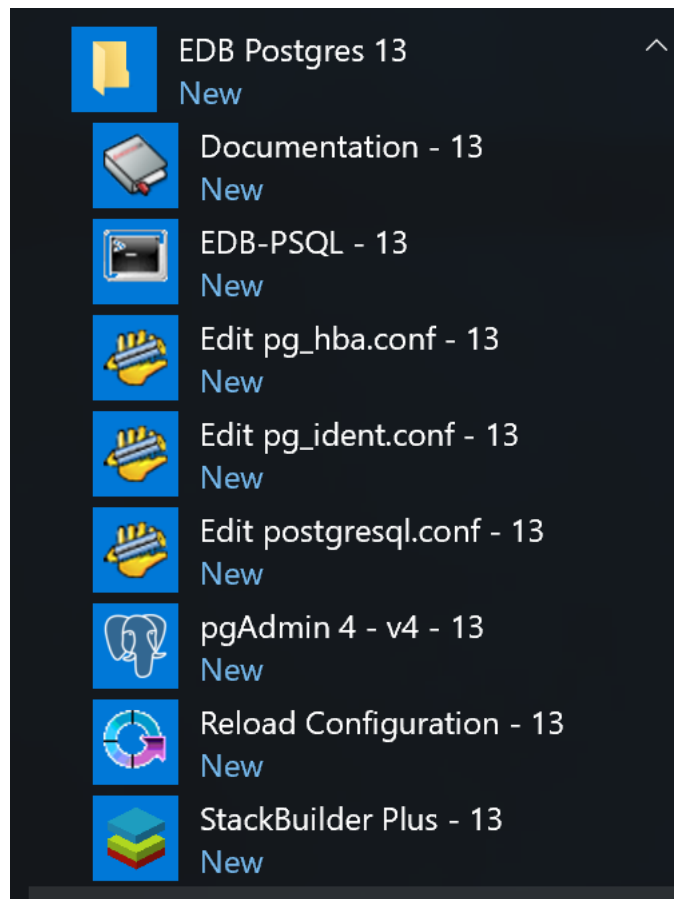


Fig. 5: Starting StackBuilder Plus

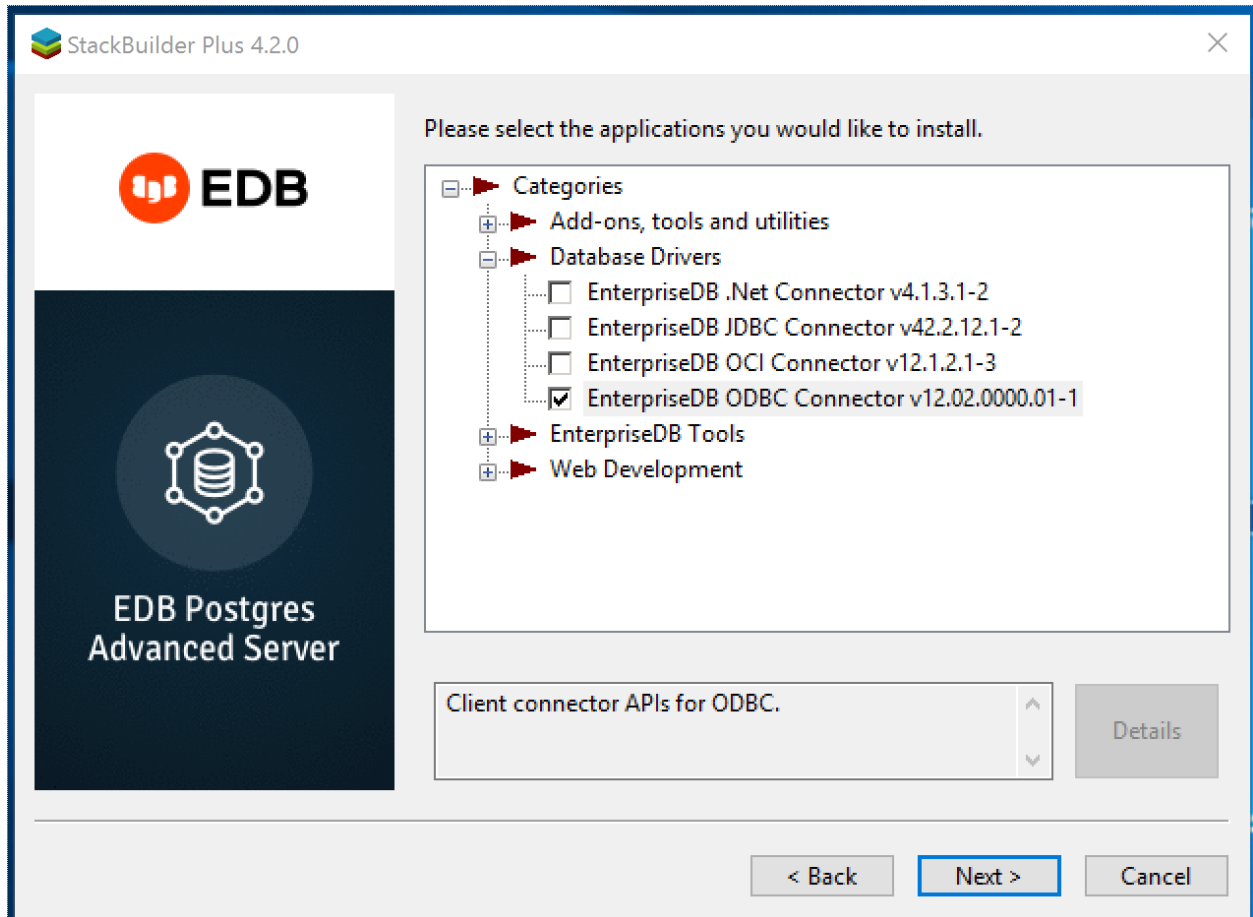


Fig. 6: Selecting the Connectors installer.

CREATING A DATA SOURCE

When a client application tries to establish a connection with a server, it typically provides a data source name (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 usually named `/etc/odbc.ini`. The name and location can 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. 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.

[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. 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.

CONNECTION PROPERTIES

The following table describes the connection properties that you can specify through the dialog boxes 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're connecting.
Driver	Driver	EDB-ODBC	The name of the ODBC driver.
Server	Servename	Localhost	The name or IP address of the server that you're connecting to.
dbms_name	dbms_name	EnterpriseDB	Database system. Either EnterpriseDB or PostgreSQL.
Description	Description		Descriptive name of the data source.
User Name	Username		The name of the user that this data source uses to connect to the server.
Password	Password		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.
Level of Rollback on Errors	Use the <i>Protocol</i> option to specify rollback behavior.	Transaction Level	Specifies how the driver handles errors: 0 - Don't roll back 1 - Roll back the transaction 2 - Roll back 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 <i>SQLTables()</i> function.
OID Options: Show Column	ShowOidColumn	No	If enabled, the <i>SQLColumns()</i> function reports the OID column.

continues on next page

Table 1 – continued from previous page

Property	Keyword name	Default value	Description
OID Options: Fake Index	FakeOidIndex	No	If enabled, the <i>SQLStatistics()</i> 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 <i>SQLStatistics()</i> function reports unique indexes. If not enabled, the <i>SQLStatistics()</i> function reports that indexes allow duplicate values.
Use Declare/Fetch	UseDeclareFetch	Off	If enabled, the driver uses server-side cursors. To enable <i>UseDeclareFetch</i> , specify a value of <i>1</i> . To disable <i>UseDeclareFetch</i> , specify a value of <i>0</i> .
CommLog	CommLog	Off	If enabled, records all client/server traffic in a log file.
Parse Statements	Parse	Off	If enabled, the driver parses simple <i>SELECT</i> statements when you call the <i>SQLNumResultCols()</i> , <i>SQLDescribeCol()</i> , or <i>SQLColAttributes()</i> functions.
Cancel as FreeStmt	CancelAsFreeStmt	Off	If enabled, the <i>SQLCancel()</i> function calls <i>SQLFreeStmt(SQL_Close)</i> on your behalf.
MyLog	Debug	Off	If enabled, the driver records its work in a log file. On Windows, the file name is <i>C:\mylog\(<>)<process-id></i> . On Linux the file name is <i>/tmp/[mylog](<>)<username><process-id>.log</i> .
Unknown Sizes	UnknownSizes	Maximum	Determines how the <i>SQLDescribeCol()</i> and <i>SQLColAttributes()</i> functions compute the size of a column. Specify <i>0</i> to force the driver to report the maximum size allowed for the type. Specify <i>1</i> to force the driver to report an unknown length or <i>2</i> to force the driver to search the result set to find the longest value. Don't specify <i>2</i> if you enabled <i>UseDeclareFetch</i> .
Text as LongVarchar	TextAsLongVarChar	8190	If enabled, the driver treats TEXT columns as if they are of type <i>SQL_LONGVARCHAR</i> . If disabled, the driver treats TEXT columns as <i>SQL_VARCHAR</i> values.

continues on next page

Table 1 – continued from previous page

Property	Keyword name	Default value	Description
Unknown as Long Varchar	LongVarChar	False	If enabled, the driver treats values of unknown type as <i>SQL_LONGVARCHAR</i> values. If disabled, the driver treats values of unknown type as <i>SQL_VARCHAR</i> values. By default, values of unknown type are treated as <i>Y</i> values.
Bools as Char	BoolsAsChar	On	If enabled, the driver treats <i>BOOL</i> columns as <i>SQL_CHAR</i> values. If disabled, <i>BOOL</i> columns are treated as <i>SQL_BIT</i> values.
Max Varchar	MaxVarcharSize	255	If enabled, the driver treats <i>VARCHAR</i> and <i>BPCHAR</i> values longer than <i>MaxVarcharSize</i> as <i>SQL_LONGVARCHAR</i> values
Max Long Varchar Size	MaxLongVarchar-Size	8190	If <i>TextAsLongVarChar</i> is on, the driver reports <i>TEXT</i> values are <i>MaxLongVarcharSize</i> bytes long. If <i>UnknownAsLongVarChar</i> is on, columns of unknown type are <i>MaxLongVarcharSize</i> bytes long. Otherwise, they are reported to be <i>MaxVarcharSize</i> bytes in length.
Cache Size	Fetch	100	Determines the number of rows fetched by the driver when <i>UseDeclareFetch</i> is enabled.
SysTable Prefixes	ExtraSysTablePrefixes	[dd](<>);	Use the <i>SysTablePrefixes</i> field to specify a semicolon-delimited list of prefixes that indicate that a table is a system table. By default, the list contains <i>[dd](<>);</i> .
Cumulative Row Count for Insert	MapSqlParcNo-Batch	Off/0	If enabled, the <i>SQLRowCount()</i> function returns a single, cumulative row count for the entire array of parameter settings for an <i>INSERT</i> statement. If disabled, an individual row count is returned for each parameter setting. By default, this option is disabled.
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 them back to line-feed characters when sending character values to the server. By default, this option is enabled.

continues on next page

Table 1 – continued from previous page

Property	Keyword name	Default value	Description
Updatable Cursors	UpdatableCursors	Off	Permits positioned <i>UPDATE</i> and <i>DELETE</i> operations using the <i>SQLSetPos()</i> or <i>SQLBulkOperations()</i> functions.
Bytea as Long VarBinary	ByteaAsLong-VarBinary	Off	If enabled, the driver treats BYTEA values as if they're of type <i>SQL_LONGVARBINARY</i> . If disabled, BYTEA values are treated as <i>SQL_VARBINARY</i> values.
Bytea as LO	ByteaAsLO	False	If enabled, the driver treats BYTEA values as if they're large objects.
Row versioning	RowVersioning	Off	The <i>Row Versioning</i> option specifies if the driver includes the <i>xmin</i> column when reporting the columns in a table. The <i>xmin</i> value is the ID of the transaction that created the row. You must use row versioning if you plan to create cursors where <i>SQL_CONCURRENCY</i> = <i>SQL_CONCUR_ROWVER</i> .
Disallow Premature	DisallowPremature	No/0	Determines driver behavior if you try to retrieve information about a query without executing the query. If <i>Yes</i> , the driver declares a cursor for the query and fetches the metadata from the cursor. If <i>No</i> , the driver executes the command as soon as you request any metadata.
True is -1	TrueIsMinus1	Off/0	<i>TrueIsMinus1</i> tells the driver to return BOOL values of TRUE as <i>-1</i> . If this option isn't enabled, the driver returns BOOL values of TRUE as <i>1</i> . The driver always returns BOOL values of FALSE as <i>0</i> .
Server side prepare	UseServerSidePrepare	No/0	If enabled, the driver uses the <i>PREPARE</i> and <i>EXECUTE</i> commands to implement the Prepare/Execute model.
Use GSSAPI for GSS request	GssAuthUseGSS	False/0	If set to <i>True/1</i> , the driver sends a GSSAPI authentication request to the server. Windows only.

continues on next page

Table 1 – continued from previous page

Property	Keyword name	Default value	Description
Int8 As	BI	0	The value of <i>BI</i> determines how the driver treats <i>BIGINT</i> values: If -5 as a <i>SQL_BIGINT</i> , If 2 as a <i>SQL_NUMERIC</i> , If 8 as a <i>SQL_DOUBLE</i> , If 4 as a <i>SQL_INTEGER</i> , If 12 as a <i>SQL_VARCHAR</i> , If 0 (on an MS Jet client), as a <i>SQL_NUMERIC</i> , If 0 on any other client, as a <i>SQL_BIGINT</i> .
Extra options Connect Settings	AB ConnSettings	0x0	0x1 - Forces the output of short-length formatted connection strings. Specify this option if you're 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 <i>SQLGetInfo</i> returns information about all tables or only accessible tables. If set, information is returned only 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 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 lower case.
Disable Genetic Optimizer	Optimizer	Yes/1	Disables the genetic query optimizer.
Allow Keyset	UpdatableCursors	Yes/1	Allow Keyset-driven cursors

continues on next page

Table 1 – continued from previous page

Property	Keyword name	Default value	Description
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 <i>SQL_TABLES</i> include only user-accessible tables.

5.1 Adding a data source definition in Windows

The Windows ODBC Data Source Administrator is a graphical interface that creates named data sources. To open the ODBC Data Source Administrator, in the Control Panel, open the **AdministrativeTools** menu and double-click the appropriate ODBC Data Sources icon (32-bit or 64-bit).

Select **Add** to open the Create New Data Source dialog box. Select **EnterpriseDB(ANSI)** or **EnterpriseDB(UNICODE)** from the list of drivers and select **Finish**.

Use the fields on the EnterpriseDB ODBC Driver dialog box to define the named data source:

- Enter the database name in the **Database** field.
- Enter the host name or IP address of EDB Postgres Advanced Server in the **Server** field.
- Enter the name of a user in the **UserName** 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 list next to the **SSLMode** label is active, allowing you to use SSL and other EDB Postgres 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.

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

Select **Global** to open a dialog where you can specify logging options for the EDB-ODBC driver (not the data source, but the driver).

- Select **DisableGeneticOptimizer** to disable the genetic query optimizer. By default, the query optimizer is on.
- Select **KSQO(KeysetQueryOptimization)** to enable server-side support for keyset queries. By default, **KeysetQueryOptimization** is on.

- Select **RecognizeUniqueIndexes** to force the `SQLStatistics()` function to report unique indexes. If the option is not selected, the `SQLStatistics()` function reports that all indexes allow duplicate values. By default, **RecognizeUniqueIndexes** is on.
- Select **UseDeclare/Fetch** to specify for the driver to use server-side cursors whenever your application executes a `SELECT` command. By default, **UseDeclare/Fetch** is off.
- Select ****CommLog(C::raw-latex:\psqlodbc_xxxx.log)**** to record all client/server traffic in a log file. By default, logging is off.
- Select **ParseStatements** to specify for the driver (rather than the server) to attempt to parse simple `SELECT` statements when you call the `SQLNumResultCols()`, `SQLDescribeCol()`, or `SQLColAttributes()` function. By default, this option is off.
- Select **CancelasFreeStmt(Exp)** to specify for the `SQLCancel()` function to call `SQLFreeStmt(SQLClose)` on your behalf. By default, this option is off.
- Select ****MyLog(C::raw-latex:\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 **UnknownSizes** box specify how the `SQLDescribeCol()` and `SQLColAttributes()` functions compute the size of a column of unknown type.

- Select **Maximum** to specify for the driver to report the maximum size allowed for a `VARCHAR` or `LONGVARCHAR` (dependent on the **UnknownsasLongVarChar** setting). If **UnknownsasLongVarChar** is enabled, the driver returns the maximum size of a `LONGVARCHAR` (specified in the **MaxLongVarChar** field in the **Miscellaneous** box). If **UnknownsasLongVarChar** is cleared, the driver returns the size specified in the **MaxVarChar** field in the **Miscellaneous** box.
- Select **Don'tknow** to specify for the driver to report a length of **unknown**.
- Select **Longest** to specify for the driver to search the result set and report the longest value found. Don't specify **Longest** if **UseDeclareFetch** is enabled.)

The properties in the **DataTypeOptions** box determine how the driver treats columns of specific types:

- Select **TextasLongVarChar** to treat `TEXT` values as if they are of type `SQL_LONGVARCHAR`. If cleared, the driver treats `TEXT` values as `SQL_VARCHAR` values. By default, `TEXT` values are treated as `SQL_LONGVARCHAR` values.
- Select **UnknownsasLongVarChar** to specify for the driver to treat values of unknown type as `SQL_LONGVARCHAR` values. If cleared, the driver treats values of unknown type as `SQL_VARCHAR` values. By default, values of unknown type are treated as `SQL_VARCHAR` values.
- Select **BoolsasChar** to specify for the driver to treat `BOOL` values as `SQL_CHAR` values. If cleared, `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 **MaxVarChar** field. By default, this value is set to 255.
- Enter the maximum length allowed for a `LONGVARCHAR` value in the **MaxLongVarChar** field. By default, this value is set to 8190.
- Specify the number of rows fetched by the driver (when **UseDeclareFetch** is enabled) in the **CacheSize** field. The default value is 100.
- Use the **SysTablePrefixes** field to specify a semicolon-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 box to their default settings by selecting **Defaults** .

Select **Apply** button to apply any changes to the data source properties. Select **OK** to apply any changes and exit.

Select **Page2** (in the upper-left corner of the Advanced Options dialog box) to access a second set of advanced options.

- Select **ReadOnly** to prevent the driver from executing the following commands: INSERT , UPDATE , DELETE , CREATE , ALTER , DROP , GRANT , REVOKE or LOCK . Invoking the **ReadOnly** option also prevents any calls that use ODBC's procedure call escape syntax (call=procedure-name?). By default, this option is off.
- Select **ShowSystemTables** to include system tables in the result set of the SQLTables() function. If the option is enabled, the driver includes 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 box. By default, this option is off.
- Select **Showsys/dboTables[Access]** to access objects in the sys schema and dbo schema through the ODBC data source. By default, this option is on.
- Select **Showsys/SQLTables** to enable accessing database tables in a linked server with MS SQL.
- Select **CumulativeRowCountforInsert** 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 cleared, 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.
- Select **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 them back to line-feed characters when sending character values to the server. By default, this option is on.
- Select **UpdatableCursors** to specify for the driver to permit positioned UPDATE and DELETE operations with the SQLSetPos() or SQLBulkOperations() functions. By default, this option is on.
- Select **byteasLO** to specify for the driver to treat BYTEA values as if they're SQL_LONGVARIABLE values. If cleared, EDB-ODBC treats BYTEA values as if they are SQL_VARIABLE values. By default, BYTEA values are treated as SQL_VARIABLE values.
- Select **RowVersioning** 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, this option is off.
- Select **DisallowPremature** to specify for the driver to retrieve metadata about a query (the number of columns in a result set or the column types) without actually executing the query. If this option is cleared, the driver executes the query when you request metadata about the query. By default, this option is off.
- Select **Trueis-1** to tell the driver to return BOOL values of True as a -1 . If this option is cleared, the driver returns BOOL values of True as 1 . The driver always returns BOOL values of False as 0 .
- Select **Serversideprepare** to tell the driver to use the PREPARE and EXECUTE commands to implement the Prepare/Execute model. By default, this option is on.
- Select **usegssapiforGSSrequest** 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 options in the **Int8As** box to specify how the driver returns BIGINT values to the client. Select **default** to specify the default type of **NUMERIC** if the client is MS Jet. Select **BIGINT** if the client is any other ODBC client. You can optionally specify for the driver to 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 **ExtraOpts** field is 0x0 . For **ExtraOpts** , you can specify the options shown in the table.

Option	Specifies
0x1	Forces the output of short-length formatted connection string. Select this option when you're using the MFC CDatabase class.
0x2	Allows MS Access to recognize PostgreSQL's serial type as AutoNumber type.
0x4	Returns 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 <i>SQLGetInfo</i> returns information about all tables or only accessible tables. If set, information is returned only 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 options 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 **LevelofRollbackonErrors** box contains options that specify how the driver handles error handling.

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

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

- Select **ShowColumn** to include the OID column in the result set of the *SQLColumns()* function. If cleared, the OID column is hidden from *SQLColumns()* .
- Select **FakeColumns** to specify for the *SQLStatistics()* function to report that a unique index exists on each OID column.

Use the **ConnectSettings** field to specify a list of parameter assignments for the driver to use when opening this connection. Any configuration parameter that you can modify with a SET statement can be included in the semicolon-delimited list. For example:

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

After you define the connection properties for the named data source, select **Apply** to apply the options. Select **OK** to save the options and exit.

Select **Global** (on the EnterpriseDB ODBC Driver dialog box) to open the Global Settings dialog box. The options on this dialog box control logging options for the EDB-ODBC driver. Use this dialog box 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.

- Select **CommLog** to record all client/server traffic in a log file. The log file is named C:\psqlodbc_<process-id> , where <process-id> is the name of the process in use.
- Select **Mylog** to keep a log file of the driver's activity. The log file is named c:\mylog_<process-id> , where <process-id> is the name of the process in use.
- Specify a location for the log files in the **Folderforlogging** field. After you entered the connection information for the named data source, select **Test** to verify that the driver manager can connect to the defined data source.

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

5.2 Adding a data source definition in Linux

The Linux ODBC Administrator is a graphical tool 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
```

```
yum install unixODBC-kde
```

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

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

If the EDB-ODBC driver doesn't appear in the list of drivers, you can add it using the ODBC Administrator. To add a driver definition, select the **Drivers** tab, and select **Add**.

In the Driver Properties window, 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`.

After you enter the driver properties for the EDB-ODBC driver, select **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. Select the **DataSourceNames** option in the left panel, and then select the appropriate DSN tab for the type of data source name you want to add:

- Select the **User** tab to add a named data source that's available only to the current user. The data source is stored in `/user/.odbc.ini`.
- Select the **System** tab to add a named data source that's available to all users. All system data sources are stored in a single file, usually `/etc/odbc.ini`.
- Select the **File** tab to add a named data source that's available to all users but that's stored in a file of your choosing.

Select the appropriate tab and select **Add**. The Create a New Data Source window opens.

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

Complete the **DataSourceProperties(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 from an ODBC application to a log file. Specify **Yes** in the **Trace** field to turn the trace utility on. 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 EDB Postgres Advanced Server database in the **Database** field.
- Enter the host name or IP address of EDB Postgres Advanced Server in the **Servename** 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 **ReadOnly** 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 you want the driver 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, **RowVersioning** is set to **No** .
- Use the **ShowSystemTables** field to specify **Yes** if you want the driver to 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 you want the driver to 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 you want the SQLStatistics() function to 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 for the driver to use when opening this connection.

After you define the connection properties, select **OK** . The new data source is added to the list of data source names.

DRIVER FUNCTIONALITY

You can use ODBC functions to query ODBC for specific information about the various attributes of the connection between EDB-ODBC and the server.

- *SQLGetInfo()* returns information about the EDB-ODBC driver and EDB Postgres Advanced Server.
- *SQLGetEnvAttr()* returns information about ODBC environment attributes.
- *SQLGetConnectAttr()* returns information about attributes specific to an individual connection.
- *SQLGetStmtAttr()* returns information about the attributes specific to an individual statement.

You can also use ODBC functions to set attributes of the objects that you use to interface with ODBC:

- Use the *SQLSetConnectAttr()* function to set connection attributes.
- Use the *SQLSetEnvAttr()* function to set environment attributes.
- Use the *SQLSetStmtAttr()* function to set statement attributes.

6.1 SQLGetInfo()

The ODBC *SQLGetInfo()* function returns information about the EDB-ODBC driver and EDB Postgres Advanced Server. You must have an open connection to call *SQLGetInfo()*, unless you specify *SQL_ODBC_VER* as the *info_type*. The signature for *SQLGetInfo()* is:

```
SQLRETURN SQLGetInfo
(
    SQLHDBC conn_handle , // Input
    SQLUSMALLINT info_type , // Input
    SQLPOINTER info_pointer , // Output
    SQLSMALLINT buffer_len , // Input
    SQLSMALLINT * string_length_pointer // Output
);
```

- *conn_handle* — The connection handle.
- *info_type* — The type of information *SQLGetInfo()* is retrieving.
- *info_pointer* — A pointer to a memory buffer to hold the retrieved value.

If the *info_type* argument is *SQL_DRIVER_HDESC* or *SQL_DRIVER_HSTMT*, the *info_pointer* argument is both Input and Output.

- *buffer_len* — The length of the allocated memory buffer pointed to by *info_pointer*. If *info_pointer* is *NULL*, *buffer_len* is ignored. If the returned value is a fixed size, *buffer_len* is ignored. *buffer_len* is used only if the requested value is returned in the form of a character string.

- *string_length_pointer* — A pointer to an *SQLSMALLINT* value. *SQLGetInfo()* writes the size of the requested value in this integer.

A typical usage is to call *SQLGetInfo()* with a *NULL info_pointer* to obtain the length of the requested value, allocate the required number of bytes, and then call *SQLGetInfo()* again (providing the address of the newly allocated buffer) to obtain the actual value. The first call retrieves the number of bytes required to hold the value. The second call retrieves the value.

If the size of the returned value exceeds *buffer_len*, the information is truncated and *NULL* terminated. If the returned value is a fixed size, *string_length* is ignored, and the size of the requested value isn't provided by *SQLGetInfo()*.

SQLGetInfo() writes information in one of the following formats:

- *SQLINTEGER* bitmask
- *SQLINTEGER* flag
- *SQLINTEGER* binary value
- *SQLSMALLINT* value
- *NULL* -terminated character string

SQLGetInfo() returns *SQL_SUCCESS*, *SQL_SUCCESS_WITH_INFO*, *SQL_ERROR*, or *SQL_INVALID_HANDLE*.

The following table lists the information returned by EDB-ODBC about the EDB Postgres Advanced Server connection.

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
<i>SQL_ACCESSIBLE_PROCEDURES</i> : Indicates if procedures returned by <i>SQLProcedures()</i> can be executed by the application.	Returns <i>N</i> . Some procedures executed by the <i>SQLProcedures()</i> function might be executed by the application.
<i>SQL_ACCESSIBLE_TABLES</i> : Indicates if the user has <i>SELECT</i> privileges on all table names returned by <i>SQLTables()</i> .	Returns <i>N</i> . The user might not have select privileges on one or more tables returned by the <i>SQLTables()</i> function.
<i>SQL_ACTIVE_CONNECTIONS</i> prev. <i>SQL_MAX_DRIVER_CONNECTIONS</i> : Indicates the maximum number of connections EDB-ODBC can support.	Returns <i>0</i> . There's no specified limit to the number of connections allowed.
<i>SQL_ACTIVE_ENVIRONMENTS</i> : The number of active environments EDB-ODBC can support.	Returns <i>0</i> . There's no specified limit to the number of environments allowed.
<i>SQL_ACTIVE_STATEMENTS</i> prev. <i>SQL_MAX_CONCURRENT_ACTIVITIES</i> : Indicates the maximum number of active statements EDB-ODBC can support.	Returns <i>0</i> . There's no specified limit to the number of active statements allowed.
<i>SQL_AGGREGATE_FUNCTION</i> : Identifies the aggregate functions supported by the server and driver.	Returns <i>SQL_AF_ALL</i> .
<i>SQL_ALTER_DOMAIN</i> : Identifies the <i>ALTER DOMAIN</i> clauses supported by the server.	Returns <i>0</i> . <i>ALTER DOMAIN</i> clauses aren't supported.
<i>SQL_ALTER_TABLE</i> : Identifies the <i>ALTER TABLE</i> clauses supported by the server.	Returns <i>SQL_AT_ADD_COLUMN</i> , <i>SQL_AT_DROP_TABLE_CONSTRAINT_CASCADE</i> , <i>SQL_AT_DROP_TABLE_CONSTRAINT</i> , <i>SQL_AT_CONSTRAINT_INITIALLY_DEFERRED</i> , <i>SQL_AT_CONSTRAINT_INITIALLY_IMMEDIATE</i> , <i>SQL_AT_CONSTRAINT_DEFERRABLE</i>
<i>SQL_ASYNC_MODE</i> : Level of asynchronous mode supported by EDB-ODBC.	Returns <i>SQL_AM_NONE</i> . Asynchronous mode isn't supported.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_BATCH_ROW_COUNT: Indicates how the driver returns row counts.	Returns <i>SQL_BRC_EXPLICIT</i> . Row counts are available when executed by calling <i>SQLExecute</i> or <i>SQLExecDirect</i> .
SQL_BATCH_SUPPORT: Indicates support for batch statement execution.	Returns <i>SQL_BS_SELECT_EXPLICIT</i> , <i>SQL_BS_ROW_COUNT_EXPLICIT</i> . The driver supports explicit batches with result set and row count generating statements.
SQL_BOOKMARK_PERSISTENCE: Indicates level of support for bookmarks.	Returns <i>SQL_BP_DELETE</i> , <i>SQL_BP_TRANSACTION</i> , <i>SQL_BP_UPDATE</i> , <i>SQL_BP_SCROLL</i> .
SQL_CATALOG_LOCATION Now SQL_QUALIFIER_LOCATION: Indicates the position of the catalog in a qualified table name.	Returns <i>SQL_CL_START</i> . The catalog portion of a qualified table name is at the beginning of the name.
SQL_CATALOG_NAME Now SQL_QUALIFIER_NAME: Indicates support for catalog names.	Returns Y. The server supports catalog names.
SQL_CATALOG_NAME_SEPARATOR Now SQL_QUALIFIER_NAME_SEPARATOR: Character separating the catalog name from the adjacent name element.	Returns '.' The server expects a '.' character between the qualifier and the table name.
SQL_CATALOG_TERM Now SQL_QUALIFIER_TERM: The term used to describe a catalog.	Returns catalog.
SQL_CATALOG_USAGE Now SQL_QUALIFIER_USAGE: Indicates the SQL statements that can refer to catalogs.	Returns <i>SQL_CU_DML_STATEMENTS</i> . Catalog names can be used in <i>SELECT</i> , <i>INSERT</i> , <i>UPDATE</i> , <i>DELETE</i> , <i>SELECT FOR UPDATE</i> and positioned <i>UPDATE</i> and <i>DELETE</i> statements.
SQL_COLLATION_SEQ: Returns the name of the collation sequence.	Returns an empty string. The name of the default collation is unknown.
SQL_COLUMN_ALIAS: Indicates server support for column aliases.	Returns Y. The server supports column aliases.
SQL_CONCAT_NULL_BEHAVIOR: Indicates how the server handles concatenation of NULL values.	Returns <i>SQL_CB_NON_NULL</i> . Concatenating a NULL value and a non-NULL value results in a NULL value.
SQL_CONVERT_BIGINT: Indicates conversion support from the <i>BIGINT</i> type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_BINARY: Indicates conversion support from the <i>BINARY</i> type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_BIT: Indicates conversion support from the <i>BIT</i> type using the <i>CONVERT</i> function.	Returns <i>SQL_CVT_INTEGER</i> , <i>SQL_CVT_BIT</i> .
SQL_CONVERT_CHAR: Indicates conversion support from the <i>CHAR</i> type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_DATE: Indicates conversion support from the <i>DATE</i> type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_DECIMAL: Indicates conversion support from the <i>DECIMAL</i> type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_DOUBLE: Indicates conversion support from the <i>DOUBLE</i> type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_FLOAT: Indicates conversion support from the <i>FLOAT</i> type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_CONVERT_FUNCTIONS: Lists the scalar conversion functions supported by the server and driver using the <i>CONVERT</i> function.	Returns <i>SQL_FN_CVT_CONVERT</i> .
SQL_CONVERT_INTEGER: Lists the conversion support from the INTEGER type using the <i>CONVERT</i> function.	Returns <i>SQL_CVT_INTEGER</i> , <i>SQL_CVT_BIT</i> .
SQL_CONVERT_INTERVAL_DAY_TIME: Indicates conversion support from the INTERVAL_DAY_TIME type using the <i>CONVERT</i> function.	This info_type isn't currently supported.
SQL_CONVERT_INTERVAL_YEAR_MONTH: Indicates conversion support from the INTERVAL_YEAR_MONTH type using the <i>CONVERT</i> function.	This info_type isn't currently supported.
SQL_CONVERT_LONGVARBINARY: Indicates conversion support for the LONG_VARBINARY type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_LONGVARCHAR: Indicates conversion support for the LONGVARCHAR type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_NUMERIC: Indicates conversion support for the NUMERIC type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_REAL: Indicates conversion support for the REAL type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_SMALLINT: Indicates conversion support for the SMALLINT type using the <i>CONVERT</i> function.	Returns <i>SQL_CVT_INTEGER</i> , <i>SQL_CVT_BIT</i> .
SQL_CONVERT_TIME: Indicates conversion support for the TIME type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CVT_TIMESTAMP: Indicates conversion support for the TIMESTAMP type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_TINYINT: Indicates conversion support for the TINYINT type using the <i>CONVERT</i> function.	Returns <i>SQL_CVT_INTEGER</i> , <i>SQL_CVT_BIT</i> .
SQL_CONVERT_VARBINARY: Indicates conversion support for the VARBINARY type using the <i>CONVERT</i> function.	Returns 0. The server doesn't support conversion.
SQL_CONVERT_VARCHAR: Indicates conversion support for the VARCHAR type using the <i>CONVERT</i> function.	Returns <i>SQL_CVT_INTEGER</i> , <i>SQL_CVT_BIT</i> .
SQL_CONVERT_WCHAR: Indicates conversion support for the WCHAR type using the <i>CONVERT</i> function.	This info_type is valid only when using the Unicode driver. Returns 0. The server doesn't support conversion.
SQL_CONVERT_WLONGVARCHAR: Indicates conversion support for the WLONGVARCHAR type using the <i>CONVERT</i> function.	This info_type is valid only when using the Unicode driver. Returns 0. The server doesn't support conversion.
SQL_CONVERT_WVARCHAR: Indicates conversion support for the WVARCHAR type using the <i>CONVERT</i> function.	This info_type is valid only when using the Unicode driver. Returns 0. The server doesn't support conversion.
SQL_CORRELATION_NAME: Indicates server support for correlation names.	Returns <i>SQL_CN_ANY</i> . Correlation names are supported and can be any valid name.
SQL_CREATE_ASSERTION: Indicates support for the <i>CREATE ASSERTION</i> statement.	Returns 0. The <i>CREATE ASSERTION</i> statement isn't supported.
SQL_CREATE_CHARACTER_SET: Indicates support for the <i>CREATE CHARACTER</i> statement.	Returns 0. The <i>CREATE CHARACTER</i> statement isn't supported.
SQL_CREATE_COLLATION: Indicates support for the <i>CREATE COLLATION</i> statement.	Returns 0. The <i>CREATE COLLATION</i> statement isn't supported.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_CREATE_DOMAIN: Indicates support for the <i>CREATE DOMAIN</i> statement.	Returns 0. The <i>CREATE DOMAIN</i> statement isn't supported.
SQL_CREATE_SCHEMA: Indicates support for the <i>CREATE SCHEMA</i> statement.	Returns <i>SQL_CS_CREATE_SCHEMA</i> , <i>SQL_CS_AUTHORIZATION</i> .
SQL_CREATE_TABLE: Indicates support for the <i>CREATE TABLE</i> statement.	Returns <i>SQL_CT_CREATE_TABLE</i> , <i>SQL_CT_GLOBAL_TEMPORARY</i> , <i>SQL_CT_CONSTRAINT_INITIALLY_DEFERRED</i> , <i>SQL_CT_CONSTRAINT_INITIALLY_IMMEDIATE</i> , <i>SQL_CT_CONSTRAINT_DEFERRABLE</i> , <i>SQL_CT_COLUMN_CONSTRAINT</i> , <i>SQL_CT_COLUMN_DEFAULT</i> , <i>SQL_CT_TABLE_CONSTRAINT</i> , <i>SQL_CT_CONSTRAINT_NAME_DEFINITION</i>
SQL_CREATE_TRANSLATION: Indicates support for the <i>CREATE TRANSLATION</i> statement.	Returns 0. The <i>CREATE TRANSLATION</i> statement isn't supported.
SQL_CREATE_VIEW: Indicates support for the <i>CREATE VIEW</i> statement.	Returns <i>SQL_CV_CREATE_VIEW</i> .
SQL_CURSOR_COMMIT_BEHAVIOR: Indicates how a <i>COMMIT</i> operation affects the cursor.	Returns <i>SQL_CB_PRESERVE</i> . Cursors are unchanged and can continue to fetch data.
SQL_CURSOR_ROLLBACK_BEHAVIOR: Indicates the server behavior after a <i>ROLLBACK</i> operation.	Returns <i>SQL_CB_PRESERVE</i> . Cursors are unchanged and can continue to fetch data.
SQL_CURSOR_SENSITIVITY: Indicates how the server synchronizes changes to a result set.	This info_type isn't currently supported.
SQL_DATA_SOURCE_NAME: Returns the server name used during connection.	The value returned is determined by the connection properties.
SQL_DATA_SOURCE_READ_ONLY: Indicates if the connection is in READ ONLY mode.	The value returned is determined by the connection properties.
SQL_DATABASE_NAME: Returns the name of the database.	The value returned is determined by the connection properties.
SQL_DATETIME_LITERALS: Indicates the DATETIME LITERALS supported by the server.	This info_type is not supported.
SQL_DBMS_NAME: Returns the name of the DBMS system.	Returns the value given by the <i>dbms_name</i> parameter from the <i>odbc.ini</i> file on Linux or the <i>dbms_name</i> field of page 2 of the Advanced Options dialog box when defining a data source in Windows. The default is <i>EnterpriseDB</i> .
SQL_DBMS_VER: Returns the server version.	Determined by the server.
SQL_DDL_INDEX: Indicates support for creating and dropping indexes.	Returns <i>SQL_DI_CREATE_INDEX</i> , <i>SQL_DI_DROP_INDEX</i> .
SQL_DEFAULT_TXN_ISOLATION: Indicates support for transaction isolation by the server.	Returns <i>TXN_READ_COMMITTED</i> . Non-repeatable or phantom reads are possible. Dirty reads aren't.
SQL_DESCRIBE_PARAMETER: Indicates support for the <i>DESCRIBE INPUT</i> statement.	Returns <i>N</i> . The <i>DESCRIBE INPUT</i> statement isn't supported.
SQL_DM_VER: The version of the driver manager.	Determined by driver manager.
SQL_DRIVER_HDBC: The driver's connection handle.	Returns an <i>SQLULEN</i> value that contains the driver's connection handle.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_DRIVER_HDESC: The driver descriptor handle.	Returns an <i>SQLULEN</i> value that contains driver's descriptor handle.
SQL_DRIVER_HENV: The driver's environment handle.	Returns an <i>SQLULEN</i> value that contains the driver's environment handle.
SQL_DRIVER_HLIB: The driver handle.	Returns an <i>SQLULEN</i> value that contains the library handle (returned to the ODBC driver manager when the manager loaded the driver).
SQL_DRIVER_HSTMT: The driver's statement handle.	Returns an <i>SQLULEN</i> value that contains the driver's statement handle.
SQL_DRIVER_NAME: The name of the driver.	Returns <i>EDB-ODBC.DLL</i>
SQL_DRIVER_ODBC_VER: Identifies the ODBC version that the driver supports.	Returns <i>03.50</i>
SQL_DRIVER_VER: Identifies the driver version.	Returns <i>9.0.0.6</i>
SQL_DROP_ASSERTION: Lists the <i>DROP ASSERTION</i> clauses supported by the server.	Returns <i>0</i> .
SQL_DROP_CHARACTER_SET: Lists the <i>DROP CHARACTER</i> clauses supported by the server.	Returns <i>0</i> .
SQL_DROP_COLLATION: Lists the <i>DROP COLLATION</i> clauses supported by the server.	Returns <i>0</i> .
SQL_DROP_DOMAIN: Lists the <i>DROP DOMAIN</i> clauses supported by the server.	Returns <i>0</i> .
SQL_DROP_SCHEMA: Lists the <i>DROP SCHEMA</i> clauses supported by the server.	Returns <i>SQL_DS_DROP_SCHEMA</i> , <i>SQL_DS_RESTRICT</i> , <i>SQL_DS_CASCADE</i> .
SQL_DROP_TABLE: Lists the <i>DROP TABLE</i> clauses supported by the server.	Returns <i>SQL_DT_DROP_TABLE</i> , <i>SQL_DS_RESTRICT</i> , <i>SQL_DS_CASCADE</i> .
SQL_DROP_TRANSLATION: Lists the <i>DROP TRANSLATION</i> clauses supported by the server.	Returns <i>0</i> .
SQL_DROP_VIEW: Lists the <i>DROP VIEW</i> clauses supported by the server.	Returns <i>SQL_DV_DROP_VIEW</i> , <i>SQL_DS_RESTRICT</i> , <i>SQL_DS_CASCADE</i> .
SQL_DYNAMIC_CURSOR_ATTRIBUTES1: Describes the first set of dynamic cursor attributes supported by the driver.	Returns <i>0</i> .
SQL_DYNAMIC_CURSOR_ATTRIBUTES2: Describes the second set of dynamic cursor attributes supported by the driver.	Returns <i>0</i> .
SQL_EXPRESSIONS_IN_ORDERBY: Indicates server support for <i>ORDER BY</i> .	Returns <i>Y</i> .
SQL_FETCH_DIRECTION: Indicates <i>FETCH</i> order options (deprecated in ODBC 3.0).	Returns <i>SQL_FD_FETCH_NEXT</i> , <i>SQL_FD_FETCH_FIRS</i> , <i>SQL_FD_FETCH_LAST</i> , <i>SQL_FD_FETCH_PRIOR</i> , <i>SQL_FD_FETCH_ABSOLUTE</i> , <i>SQL_FD_FETCH_RELATIVE</i> , <i>SQL_FD_FETCH_BOOKMARK</i> .
SQL_FILE_USAGE: Indicates how a single-tier driver treats files on the server.	Returns <i>SQL_FILE_NOT_SUPPORTED</i> . The driver isn't a single-tier file.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_FORWARD_ONLY_CURSOR_ATTRIBUTES1: Describes the forward-only cursor attributes supported by the driver.	Returns <i>SQL_CA1_NEXT</i> .
SQL_FORWARD_ONLY_CURSOR_ATTRIBUTES2: Describes extended attributes for the forward-only cursor designated by <i>SQL_FORWARD_ONLY_CURSOR_ATTRIBUTES1</i> .	Returns <i>SQL_CA2_READ_ONLY_CONCURRENCY</i> , <i>SQL_CA2_CRC_EXACT</i> .
SQL_GETDATA_EXTENSIONS: Lists supported extensions to <i>SQLGetData</i> .	Returns <i>SQL_GD_ANY_COLUMN</i> , <i>SQL_GD_ANY_ORDER</i> , <i>SQL_GD_BLOCK</i> , <i>SQL_GD_BOUND</i> .
SQL_GROUP_BY: Indicates the relationship between a <i>GROUP BY</i> clause and columns in the <i>SELECT</i> list.	Returns <i>SQL_GB_GROUP_BY_EQUALS_SELECT</i> .
SQL_IDENTIFIER_CASE: Indicates case sensitivity and case storage of SQL identifiers.	Returns <i>SQL_IC_LOWER</i> .
SQL_INDEX_KEYWORDS: Indicates support for the <i>CREATE INDEX</i> statement.	Returns <i>SQL_IK_NONE</i> .
SQL_INFO_SCHEMA_VIEWS: Lists the views supported in the <i>INFORMATION_SCHEMA</i> .	Returns 0.
SQL_INTEGRITY Prev. SQL_ODBC_SQL_OPT_IEF: Indicates server support for referential integrity syntax checking.	Returns <i>N</i> .
SQL_INSERT_STATEMENT: Indicates level of support for the <i>INSERT</i> statement.	Returns <i>SQL_IS_INSERT_LITERALS</i> , <i>SQL_IS_INSERT_SEARCHED</i> , <i>SQL_IS_SELECT_INTRO</i> .
SQL_KEYSET_CURSOR_ATTRIBUTES1: Describes the first set of keyset cursor attributes supported by the driver.	Returns <i>SQL_CA1_NEXT</i> , <i>SQL_CA1_ABSOLUTE</i> , <i>SQL_CA1_RELATIVE</i> , <i>SQL_CA1_BOOKMARK</i> , <i>SQL_CA1_LOCK_NO_CHANGE</i> , <i>SQL_CA1_POS_POSITION</i> , <i>SQL_CA1_POS_UPDATE</i> , <i>SQL_CA1_POS_DELETE</i> , <i>SQL_CA1_POS_REFRESH</i> , <i>SQL_CA1_BULK_ADD</i> , <i>SQL_CA1_BULK_UPDATE_BY_BOOKMARK</i> , <i>SQL_CA1_BULK_DELETE_BY_BOOKMARK</i> , <i>SQL_CA1_BULK_FETCH_BY_BOOKMARK</i> .
SQL_KEYSET_CURSOR_ATTRIBUTES2: Describes the second set of keyset cursor attributes supported by the driver.	Returns <i>SQL_CA2_READ_ONLY_CONCURRENCY</i> , <i>SQL_CA2_OPT_ROWVER_CONCURRENCY</i> , <i>SQL_CA2_SENSITIVITY_ADDITIONS</i> , <i>SQL_CA2_SENSITIVITY_DELETIONS</i> , <i>SQL_CA2_SENSITIVITY_UPDATES</i> , <i>SQL_CA2_CRC_EXACT</i> .
SQL_KEYWORDS: Identifies the server-specific reserved keywords.	Returns “”. There are no server-specific reserved keywords.
SQL_LIKE_ESCAPE_CLAUSE: Indicates support for an escape character in <i>LIKE</i> predicates.	Returns <i>N</i> . EDB Postgres Advanced Server doesn't support escape characters in <i>LIKE</i> predicates.
SQL_LOCK_TYPES: Lists supported lock types (deprecated in ODBC 3.0).	Returns <i>SQL_LCK_NO_CHANGE</i> .
SQL_MAX_ASYNC_CONCURRENT_STATEMENTS: The number of active concurrent statements that the driver can support.	This info_type is currently unsupported.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_MAX_BINARY_LITERAL_LEN: The maximum length of a binary literal.	Returns 0. The maximum length is unspecified.
SQL_MAX_CATALOG_NAME_LEN: The maximum length of a catalog name on the server.	Returns 0. The maximum length is unspecified.
SQL_MAX_QUALIFIER_NAME_LEN: The maximum length of a qualifier.	Returns 0. The maximum length is unspecified.
SQL_MAX_CHAR_LITERAL_LEN: The maximum number of characters in a character string.	Returns 0. The maximum length is unspecified.
SQL_MAX_COLUMN_NAME_LEN: The maximum length of a column name.	Returns 64. Column names can't exceed 64 characters in length.
SQL_MAX_COLUMNS_IN_GROUP_BY: The maximum number of columns allowed in a <i>GROUP BY</i> clause.	Returns 0. The maximum length is unspecified.
SQL_MAX_COLUMNS_IN_INDEX: The maximum number of columns allowed in an index.	Returns 0. The maximum length is unspecified.
SQL_MAX_COLUMNS_IN_ORDER_BY: The maximum number of columns allowed in an <i>ORDER BY</i> clause.	Returns 0. The maximum length is unspecified.
SQL_MAX_COLUMNS_IN_SELECT: The maximum number of columns allowed in a <i>SELECT</i> list.	Returns 0. The maximum length is unspecified.
SQL_MAX_COLUMNS_IN_TABLE: The maximum number of columns allowed in a table.	Returns 0. The maximum length is unspecified.
SQL_MAX_CONCURRENT_ACTIVITIES prev. SQL_MAX_ACTIVE_STATEMENTS: The maximum number of active SQL statements that the driver can support.	Returns 0. The maximum length is unspecified.
SQL_MAX_CURSOR_NAME_LEN: The maximum length of a cursor name.	Returns 32. A cursor name can't exceed 32 characters in length.
SQL_MAX_DRIVER_CONNECTIONS prev. SQL_ACTIVE_CONNECTIONS: The maximum number of active connections the driver can support.	Returns 0. There is no specified limit to the number of connections supported.
SQL_MAX_IDENTIFIER_LEN: The maximum identifier length allowed by the server.	Returns 64. Identifiers can't exceed 64 characters in length.
SQL_MAX_INDEX_SIZE: The maximum number of bytes allowed in the (combined) fields of an index.	Returns 0. The maximum size is unspecified.
SQL_MAX_OWNER_NAME_LEN Now SQL_MAX_SCHEMA_NAME_LEN: The maximum length of an owner name allowed by the server.	Returns 64. The maximum length of an owner name is 64 characters.
SQL_MAX_PROCEDURE_NAME_LEN: The maximum length of a procedure name allowed by the server.	Returns 0. The maximum length is unspecified.
SQL_MAX_QUALIFIER_NAME_LEN Now SQL_MAX_CATALOG_NAME_LEN: The maximum length of a qualifier name allowed by the server.	Returns 0. The maximum length of a qualifier is unspecified.
SQL_MAX_ROW_SIZE: The maximum length of a row.	Returns 0. The maximum row length is unspecified.
SQL_MAX_ROW_SIZE_INCLUDES_LONG: Indicates whether the <i>SQL_MAX_ROW_SIZE</i> includes the length of any <i>LONGVARCHAR</i> or <i>LONGVARBINARY</i> columns in the row.	Returns <i>Y</i> . <i>SQL_MAX_ROW_SIZE</i> includes the length of any <i>LONGVARCHAR</i> or <i>LONGVARBINARY</i> columns in the row.
SQL_MAX_SCHEMA_NAME_LEN: The maximum length of a schema name allowed by the server.	Returns 64. The maximum length of a schema name is 64 characters.
SQL_MAX_STATEMENT_LEN: The maximum length of a SQL statement.	Returns 0. Maximum statement length is limited by available memory.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_MAX_TABLE_NAME_LEN: The maximum length of a table name allowed by the server.	Returns 64. The maximum length of a table name is 64 characters.
SQL_MAX_TABLES_IN_SELECT: The maximum number of tables allowed in the <i>FROM</i> clause of a <i>SELECT</i> statement.	Returns 0. The maximum number of tables allowed is unspecified.
SQL_MAX_USER_NAME_LEN: The maximum length of the user name allowed by the server.	Returns 0. The maximum length of a user name is unspecified.
SQL_MULT_RESULT_SETS: Indicates server support for multiple result sets.	Returns Y. EDB Postgres Advanced Server supports multiple result sets.
SQL_MULTIPLE_ACTIVE_TXN: Indicates if the server supports multiple active transactions.	Returns Y. EDB Postgres Advanced Server supports multiple active transactions.
SQL_NEED_LONG_DATA_LEN: Indicates if the server needs the length of a <i>LONG</i> data value before receiving the value.	Returns N. EDB Postgres Advanced Server doesn't need the length of a <i>LONG</i> data value before receiving the value.
SQL_NON_NULLABLE_COLUMNS: Indicates if the server supports NOT NULL values in columns.	Returns <i>SQL_NNC_NON_NULL</i> . EDB Postgres Advanced Server does support NOT NULL values in columns.
SQL_NULL_COLLATION: Indicates where NULL values are located in a result set.	Returns <i>SQL_NC_HIGH</i> . The location of NULL values in a data set is determined by the <i>ASC</i> and <i>DESC</i> keywords. NULL values are sorted to the high end of the data set.
SQL_NUMERIC_FUNCTIONS: Lists the numeric functions supported by the driver and the server.	Returns <i>SQL_FN_NUM_ABS</i> , <i>SQL_FN_NUM_ATAN</i> , <i>SQL_FN_NUM_CEILING</i> , <i>SQL_FN_NUM_COS</i> , <i>SQL_FN_NUM_EXP</i> , <i>SQL_FN_NUM_FLOOR</i> , <i>SQL_FN_NUM_LOG</i> , <i>SQL_FN_NUM_MOD</i> , <i>SQL_FN_NUM_SIGN</i> , <i>SQL_FN_NUM_SIN</i> , <i>SQL_FN_NUM_SQRT</i> , <i>SQL_FN_NUM_TAN</i> , <i>SQL_FN_NUM_RAND</i> , <i>SQL_FN_NUM_POWER</i> , <i>SQL_FN_NUM_ROUND</i> .
SQL_ODBC_API_CONFORMANCE: Indicates the ODBC 3.0 compliance level.	Returns <i>SQL_OAC_LEVEL1</i> . The driver conforms to ODBC Level 1 interface.
SQL_ODBC_INTERFACE_CONFORMANCE: Indicates the ODBC interface that the driver adheres to.	Returns <i>SQL_OIC_CORE</i> .
SQL_ODBC_SAG_CLI_CONFORMANCE: Indicates the SQL Access Group compliance level that the driver adheres to.	Returns <i>SQL_OSCC_NOT_COMPLIANT</i> . The driver isn't SAG CLI compliant.
SQL_ODBC_SQL_CONFORMANCE: Indicates the SQL grammar level that the driver conforms to.	Returns <i>SQL_OSC_CORE</i> . The driver conforms to the core grammar level.
SQL_ODBC_SQL_OPT_IEF Now SQL_INTEGRITY: Indicates server support for referential integrity syntax checking.	Returns N. The server doesn't support referential integrity syntax checking.
SQL_ODBC_VER: The ODBC version supported by the driver manager	Returns 03.52.0000.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_OJ_CAPABILITIES: Identifies the outer joins that are supported by the server.	Returns <i>SQL_OJ_LEFT</i> , <i>SQL_OJ_RIGHT</i> , <i>SQL_OJ_FULL</i> , <i>SQL_OJ_NESTED</i> , <i>SQL_OJ_NOT_ORDERED</i> , <i>SQL_OJ_INNER</i> , <i>SQL_OJ_ALL_COMPARISON_OPS</i> .
SQL_OUTER_JOINS: Indicates support for outer joins and the outer join escape sequence.	Returns Y. Outer joins are supported.
SQL_OWNER_TERM prev. SQL_SCHEMA_TERM: The term used to describe a schema.	Returns schema.
SQL_ORDER_BY_COLUMNS_IN_SELECT: Indicates if the columns in an <i>ORDER BY</i> clause must be included in the <i>SELECT</i> list.	Returns N. Columns in an <i>ORDER BY</i> clause don't have to be in the <i>SELECT</i> list.
SQL_OWNER_USAGE prev. SQL_SCHEMA_USAGE: Returns a string that indicates which statements support schema qualifiers.	Returns <i>SQL_OU_DML_STATEMENTS</i> , <i>SQL_OU_TABLE_DEFINITION</i> , <i>SQL_OU_INDEX_DEFINITION</i> , <i>SQL_OU_PRIVILEGE_DEFINITION</i> .
SQL_PARAM_ARRAY_ROW_COUNTS: Indicates if the server returns a single row count or separate row counts for each element in an array when executing a parameterized statement with at least one parameter bound to the array.	Returns <i>SQL_PARC_BATCH</i> if separate row counts are available for each element in an array. Returns <i>SQL_PARC_NO_BATCH</i> if a single, cumulative row count is available for the entire array.
SQL_PARAM_ARRAY_SELECTS: Indicates if the server returns one result set or a separate result set for each element in an array (or if the driver doesn't allow this feature) when executing a parameterized statement with at least one parameter bound to the array.	Returns <i>SQL_PAS_BATCH</i> . One data set is available for each element in an array.
SQL_POS_OPERATION: Lists the options supported by <i>SQLSetPos()</i> .	Returns <i>SQL_POS_POSITION</i> , <i>SQL_POS_REFRESH</i> , <i>SQL_POS_UPDATE</i> , <i>SQL_POS_DELETE</i> , <i>SQL_POS_ADD</i> .
SQL_POSITIONED_STATEMENTS: Lists the supported positioned SQL statements.	Returns <i>SQL_PS_POSITIONED_DELETE</i> , <i>SQL_PS_POSITIONED_UPDATE</i> , <i>SQL_PS_SELECT_FOR_UPDATE</i> .
SQL_PROCEDURE_TERM: The term used to describe a procedure.	Returns procedure.
SQL_PROCEDURES: Indicates if the server and the driver support SQL procedures and procedure invocation syntax.	Returns Y. The server and driver support procedures and procedure invocation syntax.
SQL_QUALIFIER_LOCATION prev. SQL_CATALOG_LOCATION: Indicates the position of the schema name in a qualified table name.	Returns <i>SQL_CL_START</i> . The catalog portion of a qualified table name is at the beginning of the name.
SQL_QUALIFIER_NAME prev. SQL_CATALOG_NAME: Indicates server support for catalog names.	Returns Y. The server supports catalog names.
SQL_QUALIFIER_NAME_SEPARATOR prev. SQL_CATALOG_NAME_SEPARATOR: Character separating the qualifier name from the adjacent name element.	Returns '.'. The server expects a '.' character between the qualifier and the table name.
SQL_QUALIFIER_TERM prev. SQL_CATALOG_TERM: The term used to describe a qualifier.	Returns catalog.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_QUALIFIER_USAGE prev. SQL_CATALOG_USAGE: Indicates the SQL statements that might refer to qualifiers.	Returns <i>SQL_CU_DML_STATEMENTS</i> . Catalog names can be used in <i>SELECT</i> , <i>INSERT</i> , <i>UPDATE</i> , <i>DELETE</i> , <i>SELECT FOR UPDATE</i> , and positioned <i>UPDATE</i> and <i>DELETE</i> statements.
SQL_QUALIFIER_USAGE Now SQL_CATALOG_USAGE: Identifies DML statements that support qualifier names.	Returns <i>SQL_CU_DML_STATEMENTS</i> . Qualifiers can be used in all DML statements (<i>SELECT</i> , <i>INSERT</i> , <i>UPDATE</i> , <i>DELETE</i> , <i>SELECT FOR UPDATE</i>).
SQL_QUOTED_IDENTIFIER_CASE: Indicates case sensitivity of quoted identifiers.	Returns <i>SQL_IC_SENSITIVE</i> . Quoted identifiers are case sensitive.
SQL_CATALOG_NAME_SEPARATOR prev. SQL_QUALIFIER_NAME_SEPARATOR: The character that separates the name qualifier from the name element.	Returns ‘.’. The ‘.’ character is used as a separator in qualified names.
SQL_QUALIFIER_TERM: The term used to describe a qualifier.	Returns catalog
SQL_QUALIFIER_LOCATION: The position of the qualifier in a qualified table name.	Returns <i>SQL_CL_START</i> . The qualifier precedes the table name in a qualified table name.
SQL_ROW_UPDATES: Indicates if keyset-driven or mixed cursors maintain row versions or values.	Returns Y. Cursors maintain values for all fetched rows and can detect updates to the row values.
SQL_SCHEMA_TERM: The term used to describe a schema.	Returns schema.
SQL_SCHEMA_USAGE: Indicates the SQL statements that might refer to schemas.	Returns <i>SQL_OU_DML_STATEMENTS</i> , <i>SQL_OU_TABLE_DEFINITION</i> , <i>SQL_OU_INDEX_DEFINITION</i> , <i>SQL_OU_PRIVILEGE_DEFINITION</i> .
SQL_SCROLL_CONCURRENCY: Indicates the cursor concurrency control options supported by the server.	Returns <i>SQL_SCCO_READ_ONLY</i> , <i>SQL_SCCO_OPT_ROWVER</i> .
SQL_SCROLL_OPTIONS: Indicates the cursor scroll options supported by the server.	Returns <i>SQL_SO_FORWARD_ONLY</i> , <i>SQL_SO_KEYSET_DRIVEN</i> , <i>SQL_SO_STATIC</i> .
SQL_SEARCH_PATTERN_ESCAPE: The escape character that allows use of the wildcard characters % and _ in search patterns.	Returns ‘’. The ‘’ character is used as an escape character for the ‘%’ and ‘_’ characters in search patterns.
SQL_SERVER_NAME: Indicates the name of the host.	The returned value is determined by connection properties.
SQL_SPECIAL_CHARACTERS: Indicates any special characters allowed in identifier names.	Returns _. The underscore character is allowed in identifier names.
SQL_SQL_CONFORMANCE: Indicates the level of SQL-92 compliance.	Returns <i>SQL_SC_SQL92_ENTRY</i> . The driver is SQL92 entry-level compliant.
SQL_SQL92_DATETIME_FUNCTIONS: Lists the datetime functions supported by the server.	Returns <i>SQL_SDF_CURRENT_DATE</i> , <i>SQL_SDF_CURRENT_TIME</i> , <i>SQL_SDF_CURRENT_TIMESTAMP</i> .
SQL_SQL92_FOREIGN_KEY_DELETE_RULE: Indicates the server-enforced rules for using a foreign key in a <i>DELETE</i> statement.	Returns <i>SQL_SFKD_CASCADE</i> , <i>SQL_SFKD_NO_ACTION</i> , <i>SQL_SFKD_SET_DEFAULT</i> , <i>SQL_SFKD_SET_NULL</i> .

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_SQL92_FOREIGN_KEY_UPDATE_RULE: Indicates the server-enforced rules for using a foreign key in an <i>UPDATE</i> statement.	Returns <i>SQL_SFKU_CASCADE</i> , <i>SQL_SFKU_NO_ACTION</i> , <i>SQL_SFKU_SET_DEFAULT</i> , <i>SQL_SFKU_SET_NULL</i> .
SQL_SQL92_GRANT: Indicates the supported <i>GRANT</i> statement clauses.	Returns <i>SQL_SG_DELETE_TABLE</i> , <i>SQL_SG_INSERT_TABLE</i> , <i>SQL_SG_REFERENCES_TABLE</i> , <i>SQL_SG_SELECT_TABLE</i> , <i>SQL_SG_UPDATE_TABLE</i> .
SQL_SQL92_NUMERIC_VALUE_FUNCTIONS: Lists the scalar numeric functions supported by the server and driver.	Returns <i>SQL_SNVF_BIT_LENGTH</i> , <i>SQL_SNVF_CHAR_LENGTH</i> , <i>SQL_SNVF_CHARACTER_LENGTH</i> , <i>SQL_SNVF_EXTRACT</i> , <i>SQL_SNVF_OCTET_LENGTH</i> , <i>SQL_SNVF_POSITION</i> .
SQL_SQL92_PREDICATES: Identifies the predicates of a <i>SELECT</i> statement supported by the server.	Returns <i>SQL_SP_EXISTS</i> , <i>SQL_SP_ISNOTNULL</i> , <i>SQL_SP_ISNULL</i> , <i>SQL_SP_OVERLAPS</i> , <i>SQL_SP_LIKE</i> , <i>SQL_SP_IN</i> , <i>SQL_SP_BETWEEN</i> , <i>SQL_SP_COMPARISON</i> , <i>SQL_SP_QUANTIFIED_COMPARISON</i> .
SQL_SQL92_RELATIONAL_JOIN_OPERATORS: Identifies the relational join operators supported by the server.	Returns <i>SQL_SRJO_CROSS_JOIN</i> , <i>SQL_SRJO_EXCEPT_JOIN</i> , <i>SQL_SRJO_FULL_OUTER_JOIN</i> , <i>SQL_SRJO_INNER_JOIN</i> , <i>SQL_SRJO_INTERSECT_JOIN</i> , <i>SQL_SRJO_LEFT_OUTER_JOIN</i> , <i>SQL_SRJO_NATURAL_JOIN</i> , <i>SQL_SRJO_RIGHT_OUTER_JOIN</i> , <i>SQL_SRJO_UNION_JOIN</i> .
SQL_SQL92_REVOKE: Identifies the clauses in a <i>REVOKE</i> statement that are supported by the server.	Returns <i>SQL_SR_DELETE_TABLE</i> , <i>SQL_SR_INSERT_TABLE</i> , <i>SQL_SR_REFERENCES_TABLE</i> , <i>SQL_SR_SELECT_TABLE</i> , <i>SQL_SR_UPDATE_TABLE</i> .
SQL_SQL92_ROW_VALUE_CONSTRUCTOR: Indicates the row value constructor expressions in a <i>SELECT</i> statement that are supported by the server.	Returns <i>SQL_SRVC_VALUE_EXPRESSION</i> , <i>SQL_SRVC_NULL</i> .
SQL_SQL92_STRING_FUNCTIONS: Lists the string scalar functions supported by the server and driver.	Returns <i>SQL_SSF_CONVERT</i> , <i>SQL_SSF_LOWER</i> , <i>SQL_SSF_UPPER</i> , <i>SQL_SSF_SUBSTRING</i> , <i>SQL_SSF_TRANSLATE</i> , <i>SQL_SSF_TRIM_BOTH</i> , <i>SQL_SSF_TRIM_LEADING</i> , <i>SQL_SSF_TRIM_TRAILING</i> .
SQL_SQL92_VALUE_EXPRESSIONS: Indicates the value expressions supported by the server.	Returns <i>SQL_SVE_CASE</i> , <i>SQL_SVE_CAST</i> , <i>SQL_SVE_COALESCE</i> , <i>SQL_SVE_NULLIF</i> .
SQL_STANDARD_CLI_CONFORMANCE: Indicates the CLI standard the driver conforms to.	This info_type is currently unsupported.

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_STATIC_CURSOR_ATTRIBUTES1: Describes the first set of static cursor attributes supported by the driver.	Returns <i>SQL_CA1_NEXT</i> , <i>SQL_CA1_ABSOLUTE</i> , <i>SQL_CA1_RELATIVE</i> , <i>SQL_CA1_BOOKMARK</i> , <i>SQL_CA1_LOCK_NO_CHANGE</i> , <i>SQL_CA1_POS_POSITION</i> , <i>SQL_CA1_POS_UPDATE</i> , <i>SQL_CA1_POS_DELETE</i> , <i>SQL_CA1_POS_REFRESH</i> , <i>SQL_CA1_BULK_ADD</i> , <i>SQL_CA1_BULK_UPDATE_BY_BOOKMARK</i> , <i>SQL_CA1_BULK_DELETE_BY_BOOKMARK</i> , <i>SQL_CA1_BULK_FETCH_BY_BOOKMARK</i> .
SQL_STATIC_CURSOR_ATTRIBUTES2: Describes the second set of static cursor attributes supported by the driver.	Returns <i>SQL_CA2_READ_ONLY_CONCURRENCY</i> , <i>SQL_CA2_OPT_ROWVER_CONCURRENCY</i> , <i>SQL_CA2_SENSITIVITY_ADDITIONS</i> , <i>SQL_CA2_SENSITIVITY_DELETIONS</i> , <i>SQL_CA2_SENSITIVITY_UPDATES</i> , <i>SQL_CA2_CRC_EXACT</i> .
SQL_STATIC_SENSITIVITY: Indicates whether changes made to a static cursor by <i>SQLSetPos()</i> or <i>UPDATE</i> or <i>DELETE</i> statements are detected by the application.	Returns <i>SQL_SS_ADDITIONS</i> , <i>SQL_SS_DELETIONS</i> , <i>SQL_SS_UPDATES</i> .
SQL_STRING_FUNCTIONS: Lists the scalar string functions supported by the server and driver.	Returns <i>SQL_FN_STR_CONCAT</i> , <i>SQL_FN_STR_LTRIM</i> , <i>SQL_FN_STR_LENGTH</i> , <i>SQL_FN_STR_LOCATE</i> , <i>SQL_FN_STR_LCASE</i> , <i>SQL_FN_STR_RTRIM</i> , <i>SQL_FN_STR_SUBSTRING</i> , <i>SQL_FN_STR_UCASE</i> .
SQL_SUBQUERIES: Identifies the subquery predicates to a <i>SELECT</i> statement supported by the server.	Returns <i>SQL_SQ_COMPARISON</i> , <i>SQL_SQ_EXISTS</i> , <i>SQL_SQ_IN</i> , <i>SQL_SQ_QUANTIFIED</i> .
SQL_SYSTEM_FUNCTIONS: Lists the scalar system functions supported by the server and driver.	Returns 0.
SQL_TABLE_TERM: The term used to describe a table.	Returns table.
SQL_TIMEDATE_ADD_INTERVALS: Indicates the timestamp intervals supported by the server for the <i>TIMESTAMPADD</i> scalar function.	Returns 0.
SQL_TIMEDATE_DIFF_INTERVALS: Indicates the timestamp intervals supported by the server for the <i>TIMESTAMPDIFF</i> scalar function.	Returns 0.
SQL_TIMEDATE_FUNCTIONS: Indicates the date and time functions supported by the server.	Returns <i>SQL_FN_TD_NOW</i> , <i>SQL_FN_TD_CURDATE</i> , <i>SQL_FN_TD_CURTIME</i> .
SQL_TXN_CAPABLE: Identifies the transaction support offered by the server and driver.	Returns <i>SQL_TC_ALL</i> . Transactions can contain both DML and DDL statements.
SQL_TXN_ISOLATION_OPTION: Indicates the transaction isolation level supported by the server.	Returns <i>SQL_TXN_READ_COMMITTED</i> , <i>SQL_TXN_SERIALIZABLE</i> .

continues on next page

Table 1 – continued from previous page

SQL info_type argument and description**	EDB_ODBC/EDB Postgres Advanced Server returns
SQL_UNION: Indicates server support for the <i>UNION</i> clause.	Returns <i>SQL_U_UNION</i> , <i>SQL_U_UNION_ALL</i> .
SQL_USER_NAME: Identifies the name of the user connected to a database. Can be different than the login name.	This value is determined by the connection properties.
SQL_XOPEN_CLI_YEAR: The publication year of the X/Open specification that the driver manager complies with.	This info_type is currently unsupported.

6.2 Connection attributes

You can use the ODBC `SQLGetConnectAttr()` and `SQLSetConnectAttr()` functions to retrieve or set the value of a connection attribute.

6.2.1 SQLGetConnectAttr()

The `SQLGetConnectAttr()` function returns the current value of a connection attribute. The signature is:

```
SQLRETURN SQLGetConnectAttr
(
    SQLHDBC conn_handle, //Input
    SQLINTEGER attribute, //Input
    SQLPOINTER value_pointer, //Output
    SQLINTEGER buffer_length, //Input
    SQLINTEGER * string_length_pointer //Output
);
```

- *conn_handle* — The connection handle.
- *attribute* — Identifies the attribute whose value you want to retrieve.
- *value_pointer* — A pointer to the location in memory to receive the *attribute* value.
- *buffer_length* — If *attribute* is defined by ODBC and *value_pointer* points to a character string or binary buffer, *buffer_length* is the length of *value_pointer* . If *value_pointer* points to a fixed-size value (such as an integer), *buffer_length* is ignored.

If EDB-ODBC defines the attribute, `SQLGetConnectAttr()` sets the *buffer_length* parameter. *buffer_length* can be:

Value type	Meaning
Character string	The length of the character string
Binary buffer	The result of <i>SQL_LEN_BINARY_ATTR(length)</i>
Fixed length data type	<i>SQL_IS_INTEGER</i> or <i>SQL_IS_UINTEGER</i>
Any other type	<i>SQL_IS_POINTER</i>

- *string_length_pointer* — A pointer to a *SQLINTEGER* that receives the number of bytes available to return in *value_pointer* . If *value_pointer* is NULL, *string_length_pointer* isn't returned.

This function returns `SQL_SUCCESS` , `SQL_SUCCESS_WITH_INFO` , `SQL_NO_DATA` , `SQL_ERROR` , or `SQL_INVALID_HANDLE` .

The following table lists the connection attributes supported by EDB-ODBC.

Attribute	Supported?	Notes
SQL_ATTR_ACCESS_MODE	No	SQL_MODE_READ_WRITE
SQL_ATTR_ASYNC_ENABLE	No	SQL_ASYNC_ENABLE_OFF
SQL_ATTR_AUTO_IPD	No	
SQL_ATTR_AUTOCOMMIT	Yes	SQL_AUTOCOMMIT, SQL_AUTOCOMMIT_ON, SQL_AUTOCOMMIT_OFF
SQL_ATTR_CONNECTION_TIMEOUT	No	
SQL_ATTR_CURRENT_CATALOG	No	
SQL_ATTR_DISCONNECT_BEHAVIOR	No	
SQL_ATTR_ENLIST_IN_DTC	Yes	For win32 and with conditional compilation
SQL_ATTR_ENLIST_IN_XA	No	
SQL_ATTR_LOGIN_TIMEOUT	No	SQL_LOGIN_TIMEOUT
SQL_ATTR_ODBC_CURSORS	No	
SQL_ATTR_PACKET_SIZE	No	
SQL_ATTR_QUIET_MODE	No	
SQL_ATTR_TRACE	No	
SQL_ATTR_TRACEFILE	No	
SQL_ATTR_TRANSLATE_LIB	No	
SQL_ATTR_TRANSLATE_OPTION	No	
SQL_ATTR_TXN_ISOLATION	Yes	SQL_TXN_ISOLATION, SQL_DEFAULT_TXN_ISOLATION

6.2.2 SQLSetConnectAttr()

You can use the ODBC SQLSetConnectAttr() function to set the values of connection attributes. The signature of the function is:

```
SQLRETURN SQLSetConnectAttr
(
    SQLHDBC conn_handle , // Input
    SQLINTEGER attribute , // Input
    SQLPOINTER value_pointer , // Input
    SQLINTEGER string_length , // Input
);
```

- `conn_handle` — The connection handle.
- `attribute` — Identifies the attribute whose value you want to set.
- `value_pointer` — A pointer to the value that the attribute assumes.
- `string_length` — If `attribute` is defined by ODBC and `value_pointer` points to a binary buffer or character string, `string_length` is the length of `value_pointer` . If `value_pointer` points to a fixed-length value (such as an integer), `string_length` is ignored.

If EDB-ODBC defines the attribute, the application sets the `string_length` parameter. Possible `string_length` values are shown in the table.

Value type	Meaning
Character string	The length of the character string or <i>SQL_NTS</i>

continues on next page

Table 4 – continued from previous page

Value type	Meaning
Binary buffer	The result of <i>SQL_LEN_BINARY_ATTR(length)</i>
Fixed length data type	<i>SQL_IS_INTEGER</i> or <i>SQL_IS_UINTEGER</i>
Any other type	<i>SQL_IS_POINTER</i>

SQLSetConnectAttr() returns SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, SQL_STILL_EXECUTING or SQL_INVALID_HANDLE.

You can call SQLSetConnectAttr() any time after the connection handle is allocated, until the time that the connection is closed with a call to SQLFreeHandle(). All attributes set by the call persist until the call to SQLFreeHandle().

Connection attributes have a specific timeframe in which they can be set. Some attributes must be set before the connection is established, while others can be set only after a connection is established.

The following table lists the connection attributes and the time frame in which they can be set.

Attribute	Set before or after establishing a connection?
SQL_ATTR_ACCESS_MODE	Before or after
SQL_ATTR_ASYNC_ENABLE	Before or after
SQL_ATTR_AUTO_IPD	Before or after
SQL_ATTR_AUTOCOMMIT	Before or after
SQL_ATTR_CONNECTION_TIMEOUT	Before or after
SQL_ATTR_CURRENT_CATALOG	Before or after
SQL_ATTR_ENLIST_IN_DTC	After
SQL_ATTR_ENLIST_IN_XA	After
SQL_ATTR_LOGIN_TIMEOUT	Before
SQL_ATTR_ODBC_CURSORS	Before
SQL_ATTR_PACKET_SIZE	Before
SQL_ATTR_QUIET_MODE	Before or after
SQL_ATTR_TRACE	Before or after
SQL_ATTR_TRACEFILE	Before or after
SQL_ATTR_TRANSLATE_LIB	After
SQL_ATTR_TRANSLATE_OPTION	After
SQL_ATTR_TXN_ISOLATION	Before or after

6.3 Environment attributes

You can use the ODBC SQLGetEnvAttr() and SQLSetEnvAttr() functions to retrieve or set the value of an environment attribute.

6.3.1 SQLGetEnvAttr()

Use the SQLGetEnvAttr() function to find the current value of environment attributes on your system. The signature of the function is:

```
SQLRETURN SQLGetConnectAttr
(
    SQLHDBC env_handle, // Input
    SQLINTEGER attribute, // Input
```

(continues on next page)

(continued from previous page)

```

SQLPOINTER value_ptr, // Output
SQLINTEGER buffer_length, // Input
SQLINTEGER * string_length_pointer // Output
);

```

- `env_handle` — The environment handle.
- `attribute` — Identifies the attribute whose value you want to retrieve.
- `value_pointer` — A pointer to the location in memory to receive the attribute value.
- `buffer_length` — If the attribute is a character string, `buffer_length` is the length of `value_ptr` . If the value of the attribute isn't a character string, `buffer_length` is unused.
- `string_length_pointer` — A pointer to a `SQLINTEGER` that receives the number of bytes available to return in `value_pointer` . If `value_pointer` is `NULL`, `string_length_pointer` isn't returned.

This function returns `SQL_SUCCESS` , `SQL_SUCCESS_WITH_INFO` , `SQL_NO_DATA` , `SQL_ERROR` or `SQL_INVALID_HANDLE` .

The following table lists the environment attributes supported by EDB-ODBC.

Attribute	Supported?	Restrictions?
<code>SQL_ATTR_CONNECTION_POOLING</code>	<code>SQL_CP_ONE_PER_DRIVER</code> or <code>SQL_CP_OFF</code>	Determined by connection properties
<code>SQL_ATTR_ODBC_VERSION</code>	(<code>SQL_OV_ODBC3</code>), (<code>SQL_OV_ODBC2</code>)	None
<code>SQL_ATTR_OUTPUT_NTS</code>	<code>SQL_SUCCESS</code>	None

6.4 SQLSetEnvAttr()

You can use the `SQLSetEnvAttr()` function to set the values of environment attributes. The signature of the function is:

```

SQLRETURN SQLSetEnvAttr
(
    SQLHENV env_handle , //Input
    SQLINTEGER attribute , //Input
    SQLPOINTER value_pointer , //Input
    SQLINTEGER string_length //Input
);

```

- `env_handle` — The environment handle.
- `attribute` — Identifies the attribute whose value you want to set.
- `value_pointer` — A pointer to the value assigned to the attribute. The value is a `NULL` terminated character string or a 32-bit integer value, depending on the specified attribute.
- `string_length` — If `value_pointer` is a pointer to a binary buffer or character string, `string_length` is the length of `value_pointer` . If the value being assigned to the attribute is a character, `string_length` is the length of that character string. If `value_pointer` is `NULL`, `string_length` isn't returned. If `value_pointer` is an integer, `string_length` is ignored.

SQLSetEnvAttr() returns SQL_SUCCESS, SQL_INVALID_HANDLE, SQL_ERROR or SQL_SUCCESS_WITH_INFO. The application must call SQLSetEnvAttr() before allocating a connection handle. All values applied to environment attributes persist until SQLFreeHandle() is called for the connection. ODBC version 3.x allows you to allocate multiple environment handles simultaneously.

The following table lists the environment attributes you can set with SQLSetAttr().

Attribute	Value_pointer type	Restrictions?
SQL_ATTR_ODBC_VERSION	32-bit integer	Set this attribute before the application calls any function that includes an <i>SQLHENV</i> argument.
SQL_ATTR_OUTPUT_NTS	32-bit integer	Defaults to <i>SQL_TRUE</i> . Calls that set this attribute to <i>SQL_FALSE</i> return <i>SQL_ERROR/SQLSTATEHYC00</i> (feature not implemented).

6.5 Statement attributes

You can use the ODBC SQLGetStmtAttr() and SQLSetStmtAttr() functions to retrieve and set the value of a statement attribute.

6.5.1 SQLGetStmtAttr()

The SQLGetStmtAttr() function returns the current value of a statement attribute. The signature is:

```
SQLRETURN SQLGetConnectAttr
(
    SQLHDBC stmt_handle, //Input
    SQLINTEGER attribute, //Input
    SQLPOINTER value_ptr, //Output
    SQLINTEGER buffer_length, //Input
    SQLINTEGER * string_length_pointer //Output
);
```

- *stmt_handle* — The statement handle.
- *attribute* — The attribute value.
- *value_pointer* — A pointer to the location in memory to receive the *attribute* value.
- *buffer_length* — If the attribute is defined by ODBC, *buffer_length* is the length of *value_pointer* (if *value_pointer* points to a character string or binary buffer). If *value_pointer* points to an integer, *buffer_length* is ignored.

If EDB-ODBC defines the attribute, the application sets the *buffer_length* parameter. *buffer_length* can be:

Value type	Meaning
Character string	The length of the character string
Binary buffer	The result of <i>SQL_LEN_BINARY_ATTR(length)</i>
Fixed length data type	<i>SQL_IS_INTEGER</i> or <i>SQL_IS_UIINTEGER</i>
Any other type	<i>SQL_IS_POINTER</i>

- *string_length_pointer* — A pointer to an *SQLINTEGER* that receives the number of bytes required to hold the requested value. If *value_pointer* is NULL, *string_length_pointer* isn't returned.

This function returns SQL_SUCCESS , SQL_SUCCESS_WITH_INFO , SQL_ERROR or SQL_INVALID_HANDLE .

Attribute	Supported?	Restrictions?
SQL_ATTR_APP_PARAM_DESC	Yes	
SQL_ATTR_APP_ROW_DESC	Yes	
SQL_ATTR_ASYNC_ENABLE	No	
SQL_ATTR_CONCURRENCY	Yes	SQL_CONCUR_READ_ONLY
SQL_ATTR_CURSOR_SCROLLABLE	Yes	
SQL_ATTR_CURSOR_TYPE	Yes	SQL_CURSOR_FORWARD_ONLY
SQL_ATTR_CURSOR_SENSITIVITY	Yes	SQL_INSENSITIVE
SQL_ATTR_ENABLE_AUTO_IPD	No	
SQL_ATTR_FETCH_BOOKMARK_PTR	Yes	
SQL_ATTR_IMP_PARAM_DESC	Yes	
SQL_ATTR_IMP_ROW_DESC	Yes	
SQL_ATTR_KEYSET_SIZE	No	
SQL_ATTR_MAX_LENGTH	No	
SQL_ATTR_MAX_ROWS	No	
SQL_ATTR_METADATA_ID	Yes	
SQL_ATTR_NOSCAN	No	
SQL_ATTR_PARAM_BIND_OFFSET_PTR	Yes	ODBC V2.0
SQL_ATTR_PARAM_BIND_TYPE	Yes	
SQL_ATTR_PARAM_OPERATION_PTR	Yes	
SQL_ATTR_PARAM_STATUS_PTR	Yes	
SQL_ATTR_PARAMS_PROCESSED_PTR	Yes	
SQL_ATTR_PARAMSET_SIZE	Yes	
SQL_ATTR_QUERY_TIMEOUT	No	
SQL_ATTR_RETRIEVE_DATA	No	
SQL_ATTR_ROW_BIND_OFFSET_PTR	Yes	
SQL_ATTR_ROW_BIND_TYPE	No	
SQL_ATTR_ROW_NUMBER	Yes	
SQL_ATTR_ROW_OPERATION_PTR	Yes	
SQL_ATTR_ROW_STATUS_PTR	Yes	
SQL_ATTR_ROWS_FETCHED_PTR	Yes	
SQL_ATTR_ROW_ARRAY_SIZE	Yes	
SQL_ATTR_SIMULATE_CURSOR	No	
SQL_ATTR_USE_BOOKMARKS	Yes	
SQL_ROWSET_SIZE	Yes	

6.5.2 SQLSetStmtAttr()

You can use the `SQLSetStmtAttr()` function to set the values of environment attributes. The signature is:

```
SQLRETURN SQLSetStmtAttr
(
    SQLHENV  stmt_handle , //Input
    SQLINTEGER attribute , //Input
    SQLPOINTER value_pointer , //Input
    SQLINTEGER string_length //Input
);
```

- *stmt_handle* — The environment handle.
- *attribute* — Identifies the statement attribute whose value you want to set.
- *value_pointer* — A pointer to the location in memory that holds the value to assigned to the attribute. *value_pointer* can be a pointer to:
 - A null-terminated character string - A binary buffer - A value defined by the driver - A value of the type `SQLLEN`, `SQLULEN` or `SQLUSMALLINT`

value_pointer can also optionally hold one of the following values: - An ODBC descriptor handle - A `SQLINTEGER` value - A `SQLULEN` value - A signed `INTEGER` (if attribute is a driver-specific value)

Value type	Meaning
Character string	The length of the character string or <i>SQL_NTS</i>
Binary buffer	The result of <i>SQL_LEN_BINARY_ATTR(length)</i>
Fixed length data type	<i>SQL_IS_INTEGER</i> or <i>SQL_IS_UINTEGER</i>
Any other type	<i>SQL_IS_POINTER</i>

6.6 Error handling

You can retrieve diagnostic information for these ODBC functions mentioned by using the ODBC `SQLGetDiagRec()` function.

6.6.1 SQLGetDiagRec()

The `SQLGetDiagRec()` function returns status and error information from a diagnostic record written by the ODBC functions that retrieve or set attribute values. The signature is:

```
SQLRETURN SQLGetDiagRec
(
    SQLSMALLINT handle_type, // Input
    SQLHANDLE handle, // Input
    SQLSMALLINT record_number, // Input
    SQLCHAR *SQLState_pointer, // Output
    SQLINTEGER *native_error_pointer, // Output
    SQLCHAR *error_text_pointer, // Output
    SQLSMALLINT buffer_length, // Input
    SQLSMALLINT *text_length_pointer // Output
);
```

- *handle_type* — The handle type of the *handle* argument. *handle_type* must be one of the following:
 - `SQL_HANDLE_ENV` specifies an environment handle.
 - `SQL_HANDLE_STMT` specifies a statement handle.
 - `SQL_HANDLE_DBC` specifies a connection handle.
 - `SQL_HANDLE_DESC` specifies a descriptor handle.
- *handle* — The handle associated with the attribute error message.
- *record_number* — The status record that the application is seeking information from (must be greater than or equal to 1).
- *SQLState_pointer* — Pointer to a memory buffer that receives the *SQLState* error code from the record.
- *native_error_pointer* — Pointer to a buffer that receives the native error message for the data source (contained in the *SQL_DIAG_NATIVE* field).
- *error_text_pointer* — Pointer to a memory buffer that receives the error text (contained in the *SQL_DIAG_MESSAGE_TEXT* field).
- *buffer_length* — The length of the *error_text* buffer.
- *text_length_pointer* — Pointer to the buffer that receives the size (in characters) of the *error_text_pointer* field. If the number of characters in the *error_text_pointer* parameter exceeds the number available (in *buffer_length*), *error_text_pointer* is truncated.

SQLGetDiagRec() returns `SQL_SUCCESS`, `SQL_ERROR`, `SQL_INVALID_HANDLE`, `SQL_SUCCESS_WITH_DATA` or `SQL_NO_DATA`.

6.7 Supported ODBC API functions

The following table lists the ODBC API functions. The right column specifies Yes if the API is supported by the EDB-ODBC driver. Use the ODBC `SQLGetFunctions()` function (specifying a function ID of `SQL_API_ODBC3_ALL_FUNCTIONS`) to return a current version of this list.

ODBC API function name	Supported by EDB-ODBC?
SQLAllocConnect()	Yes
SQLAllocEnv()	Yes
SQLAllocStmt()	Yes
SQLBindCol()	Yes
SQLCancel()	Yes
SQLColAttributes()	Yes
SQLConnect()	Yes
SQLDescribeCol()	Yes
SQLDisconnect()	Yes
SQLError()	Yes
SQLExecDirect()	Yes
SQLExecute()	Yes
SQLFetch()	Yes
SQLFreeConnect()	Yes
SQLFreeEnv()	Yes
SQLFreeStmt()	Yes
SQLGetCursorName()	Yes
SQLNumResultCols()	Yes
SQLPrepare()	Yes
SQLRowCount()	Yes
SQLSetCursorName()	Yes

continues on next page

Table 11 – continued from previous page

ODBC API function name	Supported by EDB-ODBC?
SQLSetParam()	Yes
SQLTransact()	Yes
SQLColumns()	Yes
SQLDriverConnect()	Yes
SQLGetConnectOption()	Yes
SQLGetData()	Yes
SQLGetFunctions()	Yes
SQLGetInfo()	Yes
SQLGetStmtOption()	Yes
SQLGetTypeInfo()	Yes
SQLParamData()	Yes
SQLPutData()	Yes
SQLSetConnectOption()	Yes
SQLSetStmtOption()	Yes
SQLSpecialColumns()	Yes
SQLStatistics()	Yes
SQLTables()	Yes
SQLBrowseConnect()	No
SQLColumnPrivileges()	No
SQLDataSources()	Yes
SQLDescribeParam()	No
SQLExtendedFetch()	Yes
SQLForeignKeys()	Yes
SQLMoreResults()	Yes
SQLNativeSQL()	Yes
SQLNumParams()	Yes
SQLParamOptions()	Yes
SQLPrimaryKeys()	Yes
SQLProcedureColumns()	Yes
SQLProcedures()	Yes
SQLSetPos()	Yes
SQLSetScrollOptions()	No
SQLTablePrivileges()	Yes
SQLDrivers()	Yes
SQLBindParameter()	Yes
SQLAllocHandle()	Yes
SQLBindParam()	Yes
SQLCloseCursor()	Yes
SQLColAttribute()	Yes
SQLCopyDesc()	Yes
SQLEndTran()	Yes
SQLFetchScroll()	Yes
SQLFreeHandle()	Yes
SQLGetConnectAttr()	Yes
SQLGetDescField()	Yes
SQLGetDescRec()	Yes
SQLGetDiagField()	Yes
SQLGetDiagRec()	Yes
SQLGetEnvAttr()	Yes

continues on next page

Table 11 – continued from previous page

ODBC API function name	Supported by EDB-ODBC?
SQLGetStmtAttr()	Yes
SQLSetConnectAttr()	Yes
SQLSetDescField()	Yes
SQLSetDescRec()	No
SQLSetEnvAttr()	Yes
SQLSetStmtAttr()	Yes
SQLBulkOperations()	Yes

6.8 Supported data types

EDB-ODBC supports the following ODBC data types.

ODBC data type	Corresponding EDB Postgres Advanced Server data type
SQL_BIGINT	PG_TYPE_INT8
SQL_BINARY	PG_TYPE_BYTEA
SQL_BIT	PG_TYPE_BOOL or PG_TYPE_CHAR
SQL_CHAR	PG_TYPE_BPCHAR
SQL_TYPE_DATE	PG_TYPE_DATE
SQL_DECIMAL	PG_TYPE_NUMERIC
SQL_DOUBLE	PG_TYPE_FLOAT8
SQL_FLOAT	PG_TYPE_FLOAT8
SQL_INTEGER	PG_TYPE_INT4
SQL_LONGVARIABLE	PG_TYPE_BYTEA
SQL_LONGVARCHAR	PG_TYPE_VARCHAR or PG_TYPE_TEXT
SQL_NUMERIC	PG_TYPE_NUMERIC
SQL_NUMERIC	PG_TYPE_NUMERIC
SQL_REAL	PG_TYPE_FLOAT4
SQL_SMALLINT	PG_TYPE_INT2
SQL_TYPE_TIME	PG_TYPE_TIME
SQL_TYPE_TIMESTAMP	PG_TYPE_DATETIME
SQL_TINYINT	PG_TYPE_INT2
SQL_VARBINARY	PG_TYPE_BYTEA
SQL_VARCHAR	PG_TYPE_VARCHAR

6.9 Prerequisite for ADO users

You must execute `Command.Prepared = True` before executing `Command.Execute`.

6.10 Thread safety

EDB-ODBC is thread safe.

SCRAM COMPATIBILITY

The EDB ODBC Connector provides SCRAM-SHA-256 support for EDB Postgres Advanced Server versions 10 and later. This support is available from EDB ODBC 10.01.0000.01 release and later.

SLES 15 ON IBM POWER (PPC64LE)

There are two steps to completing an installation:

- Setting up the repository
- Installing the package

For each step, you must be logged in as superuser.

```
# To log in as a superuser:  
sudo su -
```

Before setting up the repository, you need to register with EDB. To receive credentials for the EDB repository, visit: [Repository Access Request](#) .

8.1 Setting up the Repository

Setting up the repository is a one time task. If you have already set up your repository, you do not need to perform these steps.

```
# Install the repository configuration and enter your EDB repository  
# credentials when prompted  
zypper addrepo https://zypp.enterprisedb.com/suse/edb-sles.repo  
  
# Install SUSEConnect to register the host with SUSE, allowing access to  
# SUSE repositories  
zypper install SUSEConnect  
  
# Register the host with SUSE, allowing access to SUSE repositories  
# Replace REGISTRATION_CODE and EMAIL with your SUSE registration  
# information  
SUSEConnect -r REGISTRATION_CODE -e EMAIL  
  
# Activate the required SUSE module  
SUSEConnect -p PackageHub/15.3/ppc64le  
  
# Refresh the metadata  
zypper refresh
```

8.2 Installing the Package

```
zypper -n install edb-odbc
```

UPGRADING A LINUX 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 Rocky Linux or AlmaLinux 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 Rocky Linux or AlmaLinux 8:

```
dnf upgrade edb-odbc
```

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