EDB*Plus

Version 41

Built at 2026-01-27T09:46:57

1 EDB*Plus

EDB*Plus is a utility program that provides a command line interface to EDB Postgres Advanced Server. EDB*Plus accepts SQL commands, SPL anonymous blocks, and EDB*Plus commands.

EDB*Plus commands are compatible with Oracle SQL*Plus commands and provide various capabilities including:

- Querying certain database objects
- Executing stored procedures
- Formatting output from SQL commands
- Executing batch scripts
- Executing OS commands
- Recording output

2 Release notes

EDB*Plus is a utility program that provides a command line interface to EDB Postgres Advanced Server.

The EDB*Plus documentation describes the latest version of EDB*Plus Version 41. The release notes provide information on what was new in each release.

Version	Release Date
41.3.0	27 Nov 2024
41.2.0	23 Aug 2023
41.1.0	20 Apr 2023
41.0.0	14 Feb 2023

2.1 EDB*Plus 41.3.0 release notes

Released: 27 Nov 2024

New features, enhancements, bug fixes, and other changes in EDB*Plus 41.3.0 include:

Type	Description	Addresses
Enhancement	EDB*Plus has been certified for use with EDB Postgres Advanced Server version 17.	
Enhancement	Enhanced the behavior where the `SET LINESIZE` command doesn't behave as expected when set to a value greater than 10. This enhancement mainly applies to the constants used in the select statement.	#103591 #35673
Bug fix	Corrected the behavior where `SPOOL` was introducing redundant blank lines after each output.	#37846
Bug fix	Fixed the issue where EDB*Plus is unable to process the command when `--` is used in between `/* */`.	#100496
Bug fix	Fixed the issue causing the execution of a trivial procedural script to perform quite slowly in EDB*Plus.	#37747
Bug fix	Fixed an issue related to the configuration of EDB*Plus with SSL certificate authentication method when the password is not specified.	#37970

2.2 EDB*Plus 41.2.0 release notes

Released: 23 Aug 2023

New features, enhancements, bug fixes, and other changes in EDB*Plus 41.2.0 include:

Type	Description
Enhancement	The connection string syntax in EDB*Plus now supports multi-host connectivity. When multiple hosts are specified, you can also use the `targetServerType` connection property and set it to `primary` to ensure that EDB*Plus always establishes a connection with the active primary EDB Postgres Advanced Server database server. [Support ticket #92553]
Bug fix	Fixed the issue whereby the ECHO command wasn't emitting the SQL statement in the spooled file when using EDB*Plus in an interactive session. [Support ticket # 83580]

2.3 EDB*Plus 41.1.0 release notes

Released: 20 Apr 2023

New features, enhancements, bug fixes, and other changes in EDB*Plus 41.1.0 include:

Type	Description
Enhancement	The `COLUMN` command in EDB*Plus now provides support for the `NEW_VALUE` parameter. The `NEW_VALUE` parameter defines a variable to hold the column value in a given EDB*Plus session.

2.4 EDB*Plus 41.0.0 release notes

Released: 14 Feb 2023

New features, enhancements, bug fixes, and other changes in EDB*Plus 41.0.0 include:

Type	Description
Enhancement	EDB*Plus now supports EDB Postgres Advanced Server 15.
Other	Starting with EDB*Plus 41 (this release), EDB*Plus is now being installed in a separate path outside of the EDB Postgres Advanced Server (EPAS) installation path. Any user processes or scripts with calls to EDB*Plus that relied on EDB*Plus being installed under EPAS specific directories, will need to be updated to use the new installation paths. On linux systems the new installation path is `/usr/edb/edbplus`. On Microsoft Windows systems, the default installation path is `C:\Program Files\edb\edbplus`.

3 Supported platforms

EDB*Plus is supported on the same platforms as EDB Postgres Advanced Server. To determine the platform support for EDB*Plus, you can either refer to the platform support for EDB Postgres Advanced Server on the Platform Compatibility page on the EDB website or refer to Installing EDB*Plus.

Supported database versions

The following list of EDB Postgres Advanced Server (EPAS) versions are currently supported for use with EDB*Plus:

- EPAS 18
- EPAS 17
- EPAS 16
- EPAS 15
- EPAS 14
- EPAS 13

4 Installing EDB*Plus

Select a link to access the applicable installation instructions:

Linux x86-64 (amd64)

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9, RHEL 8
- Oracle Linux (OL) 9, Oracle Linux (OL) 8
- Rocky Linux 9, Rocky Linux 8
- AlmaLinux 9, AlmaLinux 8

SUSE Linux Enterprise (SLES)

- SLES 15

Debian and derivatives

- Ubuntu 24.04, Ubuntu 22.04
- Debian 12, Debian 11

Linux IBM Power (ppc64le)

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9, RHEL 8

SUSE Linux Enterprise (SLES)

- SLES 15

Linux AArch64 (ARM64)

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9
- Oracle Linux (OL) 9

Debian and derivatives

- Debian 12

Windows

- Windows Server 2022 and Windows 11

4.1 Installing EDB*Plus on Linux x86 (amd64)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9
- RHEL 8
- Oracle Linux (OL) 9
- Oracle Linux (OL) 8
- Rocky Linux 9
- Rocky Linux 8
- AlmaLinux 9
- AlmaLinux 8

SUSE Linux Enterprise (SLES)

- SLES 15

Debian and derivatives

- Ubuntu 24.04
- Ubuntu 22.04
- Debian 12
- Debian 11

4.1.1 Installing EDB*Plus on RHEL 9 or OL 9 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo dnf -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.1.2 Installing EDB*Plus on RHEL 8 or OL 8 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo dnf -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.1.3 Installing EDB*Plus on AlmaLinux 9 or Rocky Linux 9 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo dnf -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.1.4 Installing EDB*Plus on AlmaLinux 8 or Rocky Linux 8 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo dnf -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.1.5 Installing EDB*Plus on SLES 15 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  zypper lr -E | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

- Activate the required SUSE module:

  ```
  sudo SUSEConnect -p PackageHub/15.7/x86_64
  ```

- Refresh the metadata:

  ```
  sudo zypper refresh
  ```

Install the package

```
sudo zypper -n install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.1.6 Installing EDB*Plus on Ubuntu 24.04 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo apt-get -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.1.7 Installing EDB*Plus on Ubuntu 22.04 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo apt-get -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.1.8 Installing EDB*Plus on Debian 12 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo apt-get -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.1.9 Installing EDB*Plus on Debian 11 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo apt-get -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.2 Installing EDB*Plus on Linux AArch64 (ARM64)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9
- Oracle Linux (OL) 9

Debian and derivatives

- Debian 12

4.2.1 Installing EDB*Plus on RHEL 9 or OL 9 arm64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo dnf -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.2.2 Installing EDB*Plus on Debian 12 arm64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo apt-get -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.3 Installing EDB*Plus on Linux IBM Power (ppc64le)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL)

- RHEL 9
- RHEL 8

SUSE Linux Enterprise (SLES)

- SLES 15

4.3.1 Installing EDB*Plus on RHEL 9 ppc64le

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo dnf -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.3.2 Installing EDB*Plus on RHEL 8 ppc64le

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
sudo dnf -y install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.3.3 Installing EDB*Plus on SLES 15 ppc64le

Prerequisites

Before you begin the installation process:

- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  zypper lr -E | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

- Activate the required SUSE module:

  ```
  sudo SUSEConnect -p PackageHub/15.7/ppc64le
  ```

- Refresh the metadata:

  ```
  sudo zypper refresh
  ```

Install the package

```
sudo zypper -n install edb-edbplus
```

Initial configuration

After performing a Linux installation of EDB*Plus, you must set the values of environment variables that allow EDB*Plus to locate your Java installation:

```
export JAVA_HOME=<path_to_java>
export PATH=<path_to_java>/bin:$PATH
```

4.4 Installing EDB*Plus on Windows

EDB provides a graphical interactive installer for Windows. You access it using StackBuilder Plus, which is installed as part of EDB Postgres Advanced Server.

Prerequisites

Before installing EDB*Plus, you must first install Java version 1.8 or later. For Windows, Java installers and instructions are available at theJava download page.

Using StackBuilder Plus

After installing EDB Postgres Advanced Server, you can use StackBuilder Plus to invoke the graphical installer for EDB*Plus. SeeUsing StackBuilder Plus.

1. Using the Windows Start menu, open StackBuilder Plus. Follow the prompts until you get to the module selection page.
2. Expand the **Add-ons, tools, and utilities** node and select **EDB*Plus**.
3. Select **Next**, which brings you to the graphical installer.

Using the graphical installer

1. Select the installation language and select **OK**.
2. On the Setup EDB*Plus page, select **Next**.
3. Browse to a directory where you want to install EDB*Plus, or leave the default location. Select **Next**.
4. On the Ready to Install page, select **Next**.

   An information box shows installation progress. Installation might take a few minutes.
5. When the installation has completed, select **Finish**.

4.5 Configuring IDENT authentication on Linux

By default, the `pg_hba.conf` file for the RPM installer enforces `IDENT` authentication. Before invoking EDB*Plus, you must either:

- Modify the `pg_hba.conf` file, changing the authentication method to a form other than `IDENT` (and restarting the server.
- Ensure that an `IDENT` server is accessible.

To ensure an `IDENT` server is accessible, you must confirm that an `identd` server is installed and running. You can use the package manager to install an `identd` server.

On RHEL/Rocky Linux/AlmaLinux 8:

```
dnf -y install xinetd authd
```

The command creates a file named `/etc/xinetd.d/auth` that contains:

```
service auth
{
     disable = yes
     socket_type = stream
wait =no
user = ident
cps = 4096 10
instances = UNLIMITED
server = /usr/sbin/in.authd server_args = -t60 --xerror -os
}
```

> **Note**
>
> If the file includes a `-E` argument at the end of the server arguments, delete `-E` .

To start the `identd` server:

```
systemctl enable xinetd
systemctl start xinetd
```

Open the `pg_ident.conf` file and create a user mapping:

```
# map_name    system_username     postgres_username
  edbas         enterprisedb        enterprisedb
```

Where:

- The name specified in the `map_name` column is a name you define to identify the mapping in the `pg_hba.conf` file.
- The name specified in the `system_username` column is `enterprisedb` .
- The name specified in the `postgres_username` column is `enterprisedb` .

Then, open the `pg_hba.conf` file and modify the `IDENT` entries:

- If you're using an IPv4 local connection, modify the file entry to read:

```
host all all 127.0.0.0/0 ident map=edbas
```

- If you're using an IPv6 local connection, modify the file entry to read:

```
host all all ::1/128 ident map=edbas
```

You must restart the EDB Postgres Advanced Server service before invoking EDB*Plus. For detailed information about EDB Postgres Advanced Server, see the EDB Postgres Advanced Server documentation.

5 Using EDB*Plus

To open an EDB*Plus command line, navigate through the **Applications** or **Start** menu to the **EDB Postgres Advanced Server** menu. Select **Run SQL Command Line > EDB*Plus**. You can also invoke EDB*Plus from the operating system command line with the following command:

```
edbplus [ -S[ILENT ] ] [ <login> | /NOLOG ] [ @<scriptfile>[.<ext> ] ]
```

`SILENT`

If specified, the EDB*Plus sign-on banner is suppressed along with all prompts.

`login`

Login information for connecting to the database server and database. `login` takes the following form. Don't use any white space in the login information.

```
<username>[/<password>][@{<connectstring> | <variable> } ]
```

`username` is a database username with which to connect to the database.

`password` is the password associated with the specified username. If you don't provide a password but a password is required for authentication, a password file is used if available. If there's no password file or no entry in the password file with the matching connection parameters, then EDB*Plus prompts for the password.

`connectstring` is the database connection string with the following format:

```
<host1>[:<port1>],<host2>[:<port2>],<host3>[:<port3>],..[/<dbname>][?ssl={true | false}]
[&targetServerType={primary}]
```

`host` is the hostname or IP address on which the database server resides. If you don't specify `@connectstring`, `@variable`, or `/NOLOG`, the default host is assumed to be the localhost.

`port` is the port number receiving connections on the database server. The default is `5444`.

> **Note**
>
> If you specify multiple hosts, the driver tries to connect once to each of them in the order specified until the connection succeeds. If none succeed, a normal connection exception is thrown. Including the `targetServerType` connection property and setting it to `primary` ensures that the connection is made only to a primary database server.

`dbname` is the name of the database to connect to. The default is `edb`.

If `Internet Protocol version 6` (IPv6) is used for the connection instead of IPv4, then the IP address must be enclosed in square brackets (that is, `[ipv6_address]`). The following is an example using an IPv6 connection:

```
edbplus.sh enterprisedb/password@[fe80::20c:29ff:fe7c:78b2]:5444/edb
```

The `pg_hba.conf` file for the database server must contain an appropriate entry for the IPv6 connection. This example shows an entry that allows all addresses:

```
# TYPE  DATABASE        USER          ADDRESS           METHOD
host    all             all           ::0/0             md5
```

For more information about the `pg_hba.conf` file, see the PostgreSQL core documentation.

If you want an SSL connection, then include the `?ssl=true` parameter in the connection string. In such a case, the connection string must minimally include `host:port`, with or without `/dbname`. The default for the `ssl` parameter is `false`. See Using a secure sockets layer (SSL) connection for instructions on setting up an SSL connection.

`variable` is a variable defined in the `login.sql` file that contains a database connection string. The `login.sql` file can be found in the `edbplus` subdirectory of the EDB Postgres Advanced Server home directory.

`/NOLOG`

Specify `/NOLOG` to start EDB*Plus without establishing a database connection. In this mode, you can't use SQL commands and EDB*Plus commands that require a database connection. You can later give the `CONNECT` command to connect to a database after starting EDB*Plus with the `/NOLOG` option.

`scriptfile[.ext ]`

`scriptfile` is the name of a file residing in the current working directory, containing SQL and/or EDB*Plus commands that execute after startup of EDB*Plus. `ext` is the filename extension. If the filename extension is `sql`, then you can omit the `.sql` extension. When creating a script file, always name the file with an extension. Otherwise EDB*Plus can't access it. EDB*Plus assumes a `.sql` extension on filenames that are specified with no extension.

> **Note**
>
> When you run the commands in the following examples you may be using a newer version of EDB*Plus and as such the EDB*Plus build number shown in your output may be different.

The following example shows user `enterprisedb` with password `password` connecting to database `edb` running on a database server on the localhost at port 5444.

```
C:\Program Files\edb\edbplus>edbplus enterprisedb/password
Connected to EnterpriseDB 16.4.1 (localhost:5444/edb) AS enterprisedb

EDB*Plus: (Build 41.3.0)
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.

SQL>
```

The following example shows user `enterprisedb` with password `password` connecting to database `edb` running on a database server on the localhost at port 5445.

```
C:\Program Files\edb\edbplus>edbplus enterprisedb/password@localhost:5445/edb
Connected to EnterpriseDB 16.4.1 (localhost:5445/edb) AS enterprisedb

EDB*Plus: (Build 41.3.0)
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.

SQL>
```

Using variable `hr_5445` in the `login.sql` file, the following shows how it is used to connect to database `hr` on localhost at port 5445.

```
C:\Program Files\edb\edbplus>edbplus enterprisedb/password@hr_5445
Connected to EnterpriseDB 16.4.1 (localhost:5445/hr) AS enterprisedb

EDB*Plus: (Build 41.3.0)
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.

SQL>
```

The following is the content of the `login.sql` file used in this example.

```
define
edb="localhost:5445/edb"
define hr_5445="localhost:5445/hr"
```

You can also define multi-host database connection strings in the `login.sql` file with the `?targetServerType=primary` parameter included in the connection string. The following shows how you can define a multi-host connection string in `login.sql`:

```
define
edb="192.168.2.24:5444,192.168.2.25:5445,192.168.2.26:5446/edb"
```

The following example executes a script file, `dept_query.sql`, after connecting to database `edb` on server localhost at port 5444.

```
C:\Program Files\edb\edbplus>edbplus enterprisedb/password
@dept_query
Connected to EnterpriseDB 16.4.1 (localhost:5444/edb) AS enterprisedb

SQL> SELECT * FROM dept;

DEPTNO DNAME
LOC
------ -------------- ------------
-
10     ACCOUNTING     NEW YORK
20     RESEARCH
DALLAS
30     SALES          CHICAGO
40     OPERATIONS
BOSTON

SQL> EXIT
Disconnected from EnterpriseDB Database.
```

The following is the content of file `dept_query.sql` used in this example.

```
SET PAGESIZE
9999
SET ECHO ON
SELECT * FROM dept;
EXIT
```

6 Using an SSL connection

EDB*Plus can connect to the EDB Postgres Advanced Server database using secure sockets layer (SSL) connectivity.

Using SSL requires various prerequisite configuration steps performed on the database server involved with the SSL connection as well as creating the Java truststore and keystore on the host that runs EDB*Plus.

The Java *truststore* is the file containing the certificate authority (CA) certificates. The Java client (EDB*Plus) uses the certificate to verify the authenticity of the server to which it is initiating an SSL connection.

The Java *keystore* is the file containing private and public keys and their corresponding certificates. The keystore is required for client authentication to the server, which is used for the EDB*Plus connection.

Refer to this material for guidance in setting up the SSL connections:

- For information on setting up SSL connectivity to the EDB Postges Advanced Server database, see Secure TCP/IP Connections with SSL in the PostgreSQL core documentation.
- For information on JDBC client connectivity using SSL, see Configuring the Client in the PostgreSQL JDBC Interface documentation.

Configuring SSL on EDB Postgres Advanced Server

This example configures SSL on a database server to show the use of SSL with EDB*Plus. A self-signed certificate is used for this purpose.

Step 1: Create the certificate signing request (CSR).

In the following example, the generated certificate signing request file is `server.csr`. The private key is generated as file `server.key`.

```
$ openssl req -new -nodes -text -out server.csr \
> -keyout server.key -subj "/CN=enterprisedb"
Generating a 2048 bit RSA private key
..........................+++
.....................................................................+++
writing new private key to 'server.key'
-----
```

> **Note**
>
> When creating the certificate, the value specified for the common name field (`CN=enterprisedb` in this example) must be the host name that is specified when connecting to EDB*Plus.

In addition, you can use user name maps as defined in the `pg_ident.conf` file to permit more flexibility for the common name and database user name, described in later steps.

Step 2: Generate the self-signed certificate.

The following generates a self-signed certificate to file `server.crt` using the certificate signing request file, `server.csr`, and the private key, `server.key`, as input.

```
$ openssl x509 -req -days 365 -in server.csr -signkey server.key \
> -out server.crt
Signature ok
subject=/CN=enterprisedb
Getting Private key
```

Step 3: Make a copy of the server certificate (`server.crt`) to use as the root certificate authority (CA) file (`root.crt`).

```
$ cp server.crt root.crt
```

Step 4: Delete the now redundant certificate signing request (`server.csr`).

```
$ rm -f server.csr
```

Step 5: Move or copy the certificate and private key files to the EDB Postgres Advanced Server data directory (for example, `/var/lib/edb/as15/data`).

```
$ mv root.crt /var/lib/edb/as15/data
$ mv server.crt /var/lib/edb/as15/data
$ mv server.key /var/lib/edb/as15/data
```

Step 6: Set the file ownership and permissions on the certificate files and private key file.

Set the ownership to the operating system account that owns the data subdirectory of the database server. Set the permissions so that no groups or accounts other than the owner can access these files.

```
$ chown enterprisedb root.crt server.crt server.key
$ chgrp enterprisedb root.crt server.crt server.key
$ chmod 600 root.crt server.crt server.key
$ ls -l
total 152
        .
        .
        .
-rw------- 1 enterprisedb enterprisedb 985 Aug 22 11:00 root.crt
-rw------- 1 enterprisedb enterprisedb 985 Aug 22 10:59 server.crt
-rw------- 1 enterprisedb enterprisedb 1704 Aug 22 10:58 server.key
```

Step 7: In the `postgresql.conf` file, make the following changes.

```
ssl = on
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
ssl_ca_file = 'root.crt'
```

Step 8: Modify the `pg_hba.conf` file to enable SSL use on the database to which you want EDB*Plus to make the SSL connection.

In the `pg_hba.conf` file, the `hostssl` type indicates the entry is used to validate SSL connection attempts from the client (EDB*Plus).

The authentication method is set to `cert` with the option `clientcert=verify-full` . This setting requires an SSL certificate from the client against which authentication is performed using the common name of the certificate (`enterprisedb` in this example).

The `map=sslusers` option specifies to use a mapping named `sslusers` defined in the `pg_ident.conf` file for authentication. This mapping allows a connection to the database if the common name from the certificate and the database user name attempting the connection match the `SYSTEM-USERNAME/PG-USERNAME` pair listed in the `pg_ident.conf` file.

The following is an example of the settings in the `pg_hba.conf` file if the database (`edb`) must use SSL connections.

```
# TYPE  DATABASE      USER          ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all           all                                   md5
# IPv4 local connections:
hostssl   edb         all  192.168.2.0/24   cert clientcert=verify-full map=sslusers
```

Step 9: The following shows the username maps in the `pg_ident.conf` file related to the `pg_hba.conf` file by the `map=sslusers` option. These username maps permit you to specify database user names `edbuser` , `postgres` , or `enterprisedb` when connecting with EDB*Plus.

```
# MAPNAME   SYSTEM-USERNAME   PG-USERNAME
 sslusers   enterprisedb      edbuser
 sslusers   enterprisedb      postgres
 sslusers   enterprisedb      enterprisedb
```

Step 10: Restart the database server.

Configuring SSL for the EDB*Plus client

After you configure SSL on the database server, this example shows how to generate certificate and keystore files for EDB*Plus (the JDBC client).

Step 1: Using files `server.crt` and `server.key` located under the database server data subdirectory, create copies of these files and move them to the host for EDB*Plus.

Store these files in the directory to contain the trusted certificate and keystore files you generate. The suggested location is to create a `.postgresql` subdirectory under the home user account that invokes EDB*Plus. Thus, these files are under the `~/.postgresql` directory of the user account that runs EDB*Plus.

For this example, assume file `edb.crt` is a copy of `server.crt` and `edb.key` is a copy of `server.key` .

Step 2: Create an additional copy of `edb.crt` .

```
$ cp edb.crt edb_root.crt
$ ls -l
total 12
-rw-r--r-- 1 user user  985   Aug 22 14:17  edb.crt
-rw-r--r-- 1 user user  1704  Aug 22 14:18  edb.key
-rw-r--r-- 1 user user  985   Aug 22 14:19  edb_root.crt
```

Step 3: Create a distinguished encoding rules (DER) format of file `edb_root.crt` . The generated DER format of this file is `edb_root.crt.der` . The DER format of the file is required for the `keytool` program used next.

```
$ openssl x509 -in edb_root.crt -out edb_root.crt.der -outform der
$ ls -l
total 16
-rw-r--r-- 1 user user  985  Aug 22 14:17  edb.crt
-rw-r--r-- 1 user user  1704 Aug 22 14:18  edb.key
-rw-r--r-- 1 user user  985  Aug 22 14:19  edb_root.crt
-rw-rw-r-- 1 user user  686  Aug 22 14:21  edb_root.crt.der
```

Step 4: Use the `keytool` program to create a keystore file `(postgresql.keystore)` using `edb_root.crt.der` as the input. This process adds the certificate of the Postgres database server to the keystore file.

> **Note**
>
> The file name `postgresql.keystore` is recommended so that you can access it in its default location `~/.postgresql postgresql.keystore`, which is under the home directory of the user account invoking EDB*Plus. The file name suffix can be `.jks` instead of `.keystore` (that is, `postgresql.jks`). In these examples, the file name `postgresql.keystore` is used.

For Windows only: The path is `%APPDATA%\.postgresql\postgresql.keystore`

The `keytool` program can be found under the `bin` subdirectory of the Java Runtime Environment installation.

You are prompted for a new password. Save this password as you must specify it with the `PGSSLCERTPASS` environment variable.

```
$ /usr/java/jdk1.8.0_131/jre/bin/keytool -keystore postgresql.keystore \
> -alias postgresqlstore -import -file edb_root.crt.der
Enter keystore password:
Re-enter new password:
Owner: CN=enterprisedb
Issuer: CN=enterprisedb
Serial number: c60f40256b0e8d53
Valid from: Tue Aug 22 10:59:25 EDT 2017 until: Wed Aug 22 10:59:25 EDT 2018
Certificate fingerprints:
      MD5:  85:0B:E9:A7:6E:4F:7C:B0:9B:D6:3A:44:55:E2:E9:8E
      SHA1: DD:A6:71:24:0B:6C:F8:BC:7A:4C:89:9B:DC:22:6A:6C:B0:F5:3F:7C
      SHA256:
DC:02:64:E2:B0:E9:6F:1C:FC:4F:AE:E6:18:85:0B:79:57:43:C3:C5:AE:43:0D:37
:49:53:6D:11:69:06:46:48
      Signature algorithm name: SHA1withRSA
      Version: 1
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Step 5: Create a `PKCS #12` format of the keystore file `(postgresql.p12)` using files `edb.crt` and `edb.key` as input.

> **Note**
>
> The file name `postgresql.p12` is recommended so that you can access it in its default location `~/.postgresql/postgresql.p12`, which is under the home directory of the user account invoking EDB*Plus.

For Windows only: The path is `%APPDATA%\.postgresql\postgresql.p12`

You're prompted for a new password. Save this password as you must specify it with the `PGSSLKEYPASS` environment variable.

```
$ openssl pkcs12 -export -in edb.crt -inkey edb.key -out postgresql.p12
Enter Export Password:
Verifying - Enter Export Password:
$ ls -l
total 24
-rw-rw-r-- 1 user user  985 Aug 24 12:18 edb.crt
-rw-rw-r-- 1 user user 1704 Aug 24 12:18 edb.key
-rw-rw-r-- 1 user user  985 Aug 24 12:20 edb_root.crt
-rw-rw-r-- 1 user user  686 Aug 24 12:20 edb_root.crt.der
-rw-rw-r-- 1 user user  758 Aug 24 12:26 postgresql.keystore
-rw-rw-r-- 1 user user 2285 Aug 24 12:28 postgresql.p12
```

Step 6: If the `postgresql.keystore` and `postgresql.p12` files aren't already in the `~/.postgresql` directory, move or copy them to that location.

For Windows only: The directory is `%APPDATA%\.postgresql`

Step 7: If the default location `~/.postgresql` isn't used, then you must set the full path (including the file name) to the `postgresql.keystore` file with the `PGSSLCERT` environment variable. You must also set the full path (including the file name) to file `postgresql.p12` with the `PGSSLKEY` environment variable before invoking EDB*Plus.

In addition, if the generated file from Step 4 wasn't named `postgresql.keystore` or `postgresql.jks`, then use the `PGSSLCERT` environment variable to set the file name and its location. Similarly, if the generated file from Step 5 wasn't named `postgresql.p12`, then use the `PGSSLKEY` environment variable to set the file name and its location.

Requesting an SSL connection between EDB*Plus and the EDB Postgres Advanced Server database

To perform an SSL connection, be sure to address the following:

- The trusted certificate and keystore files were generated for both the database server and the client host invoking EDB*Plus.
- The `postgresql.conf` file for the database server contains the updated configuration parameters.
- The `pg_hba.conf` file for the database server contains the required entry for permitting the SSL connection.
- For the client host, either the client's certificate and keystore files were placed in the user account's `~/.postgresql` directory or the environment variables `PGSSLCERT` and `PGSSLKEY` were set before invoking EDB*Plus.
- The `PGSSLCERTPASS` environment variable is set with a password.
- The `PGSSLKEYPASS` environment variable is set with a password

When invoking EDB*Plus, include the `?ssl=true` parameter in the database connection string as shown for the `connectstring` option in Using EDB*Plus.

The following is an example in which EDB*Plus is invoked from a host that's remote to the database server.

The `postgresql.conf` file of the database server contains the following modified parameters:

```
ssl = on
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
ssl_ca_file = 'root.crt'
```

The `pg_hba.conf` file of the database server contains the following entry for connecting from EDB*Plus on the remote host:

```
# TYPE  DATABASE        USER          ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                   md5
# IPv4 local connections:
hostssl   edb           all           192.168.2.24/32 cert clientcert=verify-full
```

On the remote host where EDB*Plus is invoked, the Linux user account named `user` contains the certificate and keystore files in its `~/.postgresql` directory:

```
[user@localhost ~]$ whoami
user
[user@localhost ~]$ cd .postgresql
[user@localhost .postgresql]$ pwd
/home/user/.postgresql
[user@localhost .postgresql]$ ls -l
total 8
-rw-rw-r-- 1 user user  758 Aug 24 12:37 postgresql.keystore
-rw-rw-r-- 1 user user 2285 Aug 24 12:37 postgresql.p12
```

Logged into Linux with the account named `user` , EDB*Plus is successfully invoked with the `ssl=true` parameter:

```
$ export PGSSLCERTPASS=keypass
$ export PGSSLKEYPASS=exppass
$ cd /usr/edb/edbplus
$ ./edbplus.sh enterprisedb/password@192.168.2.22:5444/edb?ssl=true
Connected to EnterpriseDB 16.4.1 (192.168.2.22:5444/edb) AS enterprisedb

EDB*Plus: (Build 41.3.0)
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.

SQL>
```

Alternatively, without placing the certificate and keystore files in `~/.postgresql` but in a different directory, you can invoke EDB*Plus in the following manner:

```
$ export PGSSLCERT=/home/user/ssl/postgresql.keystore
$ export PGSSLKEY=/home/user/ssl/postgresql.p12
$ export PGSSLCERTPASS=keypass
$ export PGSSLKEYPASS=exppass
$ cd /usr/edb/edbplus
$ ./edbplus.sh enterprisedb/password@192.168.2.22:5444/edb?ssl=true
Connected to EnterpriseDB 16.4.1 (192.168.2.22:5444/edb) AS enterprisedb

EDB*Plus: (Build 41.3.0)
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.

SQL>
```

In both cases the database user name used to log into EDB*Plus is `enterprisedb` , as this is the user specified for the common name field when creating the certificate in Step 1 of Configuring SSL on EDB Postgres Advanced Server.

You can use other database user names if the `pg_hba.conf` file with the `map` option and the `pg_ident.conf` file are used as described in Steps 8 and 9 of Configuring SSL on EDB Postgres Advanced Server.

7 Command summary

Use these commands with EDB*Plus.

ACCEPT

The `ACCEPT` command displays a prompt and waits for keyboard input. The value from the input is placed in the specified variable.

```
ACC[EPT ]
variable
```

This example creates a new variable named `my_name`, accepts a value of `John Smith`, and then displays the value using the `DEFINE` command.

```
SQL> ACCEPT
my_name
Enter value for my_name: John
Smith
SQL> DEFINE
my_name
DEFINE MY_NAME = "John
Smith"
```

APPEND

`APPEND` appends the given text to the end of the current line in the SQL buffer.

```
A[PPEND ]
text
```

In this example, a `SELECT` command is built in the SQL buffer using the `APPEND` command. Two spaces are placed between the `APPEND` command and the `WHERE` clause to separate `dept` and `WHERE` by one space in the SQL buffer.

```
SQL> APPEND SELECT * FROM
dept
SQL> LIST
  1 SELECT * FROM dept
SQL> APPEND WHERE deptno =
10
SQL> LIST
  1 SELECT * FROM dept WHERE deptno =
10
```

CHANGE

`CHANGE` performs a search-and-replace on the current line in the SQL buffer.

```
C[HANGE ] FROM [ TO
]
```

If you specify `TO/`, the first occurrence of the text `FROM` in the current line is changed to text `TO`. If you omit `TO/`, the first occurrence of the text `FROM` in the current line is deleted.

This sequence of commands makes line 3 the current line and then changes the department number in the `WHERE` clause from 20 to 30.

```
SQL> LIST
  1  SELECT empno, ename, job, sal,
comm
  2  FROM
emp
  3  WHERE deptno =
20
  4* ORDER BY empno
SQL> 3
  3* WHERE deptno =
20
SQL> CHANGE /20/30/
  3* WHERE deptno =
30
SQL> LIST
  1  SELECT empno, ename, job, sal,
comm
  2  FROM
emp
  3  WHERE deptno =
30
  4* ORDER BY empno
```

CLEAR

The `CLEAR` command removes the contents of the SQL buffer, deletes all column definitions set with the `COLUMN` command, or clears the screen.

```
CL[EAR ] [ BUFF[ER ] | SQL | COL[UMNS ] | SCR[EEN ]
]
```

`BUFFER | SQL`

Clears the SQL buffer.

`COLUMNS`

Removes column definitions.

`SCREEN`

Clears the screen. This is the default.

COLUMN

The `COLUMN` command controls output formatting. The formatting attributes set by using the `COLUMN` command remain in effect only for the current session.

```
COL[UMN] [column_name [CLE[AR]
|
                       FOR[MAT] format_spec
|
                       HEA[DING] heading_text
|
                       NEW_V[ALUE] variable
|
                       ON
|
                       OFF...]]
```

If you specify the `COLUMN` command with no other options, formatting options for current columns in effect for the session are displayed.

If the `COLUMN` command is followed by a column name, then the column name can be followed by one of the following:

- No other options
- `CLEAR`
- Any combination of `FORMAT`, `HEADING`, `NEW_VALUE` and either `OFF` or `ON`

`column_name`

Name of a column in a table to which column formatting options apply. If no other options follow `column_name`, then the current column formatting options of `column_name`, if any, are displayed.

`CLEAR`

The `CLEAR` option reverts all formatting options to their defaults for `column_name`. If you specify the `CLEAR` option, it must be the only option specified.

`format_spec`

Format specification to apply to `column_name`. For character columns, `format_spec` takes the following format:

`n`

`n` is a positive integer that specifies the column width in characters within which to display the data. Data in excess of `n` wraps around with the specified column width.

For numeric columns, `format_spec` is made up of the following elements.

Element	Description
`$`	Display a leading dollar sign.
`,`	Display a comma in the indicated position.
`.`	Marks the location of the decimal point.
`0`	Display leading zeros.
`9`	Number of significant digits to display.

If loss of significant digits occurs due to overflow of the format, then all #s are displayed.

`heading_text`

Text to use for the column heading of `column_name`.

`NEW_V[ALUE]`

This variable option is valid if specified alone or with other already supported options. A variable specified in the `COLUMN` command option is associated with the given column. The variable can contain the data value of a column or expression of a variable in the SQL `SELECT` list. You can use it in the SQL script for the rest of the session.

To contain the `departmentnumber` value for the `deptno` column, you can define a variable using the `NEW_VALUE` option:

```
SQL> column deptno new_value
departmentnumber
SQL> select deptno,dname from dept;

DEPTNO DNAME
------ --------------
-
    10 ACCOUNTING
    20 RESEARCH
    30 SALES
    40 OPERATIONS

SQL> prompt departmentnumber:
'&departmentnumber'
departmentnumber: '40'
SQL> INSERT INTO emp VALUES (1,'micheal','tester',7902,'17-DEC-
80',800,NULL,'&departmentnumber');

1 row INSERTED.
```

The `NEW_V[ALUE]` variable has the following limitations:

- Column values differ when they're seeded using the `NEW_VALUE` parameter for the "time with time zone" data type.
- The `NEW_VALUE` variable prompts for the value if the corresponding column value is `NULL`. Oracle treats `NULL` and an empty string the same, while EDB Postgres Advanced Server has a different behavior. Therefore, in EDB*Plus it prompts for the value.
- The variable value doesn't show in the `COLUMN` header if you reference the `NEW_VALUE` variable in a `SELECT` query without an alias.

`OFF | ON`

If you specify `OFF`, formatting options revert to their defaults but are still available in the session. If you specify `ON`, the formatting options specified by previous `COLUMN` commands for `column_name` in the session are reactivated.

This example shows the effect of changing the display width of the `job` column:

```
SQL> SET PAGESIZE
9999
SQL> COLUMN job FORMAT
A5
SQL> COLUMN
job
```

```
COLUMN   JOB
ON
FORMAT   A5
wrapped
SQL> SELECT empno, ename, job FROM
emp;
```

output

```
EMPNO ENAME      JOB
----- ---------- -----
 7369 SMITH      CLERK
 7499 ALLEN      SALES
                 MAN

 7521 WARD       SALES
                 MAN

 7566 JONES      MANAG
                 ER

 7654 MARTIN     SALES
                 MAN

 7698 BLAKE      MANAG
                 ER

 7782 CLARK      MANAG
                 ER

 7788 SCOTT      ANALY
                 ST

 7839 KING       PRESI
                 DENT

 7844 TURNER     SALES
                 MAN

 7876 ADAMS      CLERK
 7900 JAMES      CLERK
 7902 FORD       ANALY
                 ST

 7934 MILLER     CLERK

14 rows retrieved.
```

This example applies a format to the `sal` column:

```
SQL> COLUMN sal FORMAT
$99,999.00
SQL> COLUMN
COLUMN   JOB
ON
FORMAT   A5
wrapped
```

```
COLUMN   SAL
ON
FORMAT
$99,999.00
wrapped
SQL> SELECT empno, ename, job, sal FROM
emp;
```

output

```
EMPNO ENAME      JOB           SAL
----- ---------- ----- -----------
 7369 SMITH      CLERK     $800.00
 7499 ALLEN      SALES   $1,600.00
                 MAN

 7521 WARD       SALES   $1,250.00
                 MAN

 7566 JONES      MANAG   $2,975.00
                 ER

 7654 MARTIN     SALES   $1,250.00
                 MAN

 7698 BLAKE      MANAG   $2,850.00
                 ER

 7782 CLARK      MANAG   $2,450.00
                 ER

 7788 SCOTT      ANALY   $3,000.00
                 ST

 7839 KING       PRESI   $5,000.00
                 DENT

 7844 TURNER     SALES   $1,500.00
                 MAN

 7876 ADAMS      CLERK   $1,100.00
 7900 JAMES      CLERK     $950.00
 7902 FORD       ANALY   $3,000.00
                 ST

 7934 MILLER     CLERK   $1,300.00

14 rows retrieved.
```

CONNECT

Change the database connection to a different user or connect to a different database. There must be no white space between any of the parameters following the `CONNECT` command. The syntax is:

```
CON[NECT] <username>[/<password>][@{<connectstring> | <variable> }
]
```

Where:

`username` is a database username with which to connect to the database.

`password` is the password associated with the specified username. If you don't provide a password, but a password is required for authentication, a search is made for a password file. The search looks first in the home directory of the Linux operating system account invoking EDB*Plus (or in the `%APPDATA%\postgresql\` directory for Windows) and then at the location specified by the `PGPASSFILE` environment variable. The password file is `.pgpass` on Linux hosts and `pgpass.conf` on Windows hosts. The following is an example on a Windows host:

```
C:\Users\Administrator\AppData\Roaming\postgresql\pgpass.conf
```

If a password file can't be located or it doesn't have an entry matching the EDB*Plus connection parameters, then EDB*Plus prompts for the password. For more information about password files, see the PostgreSQL core documentation.

!!! Note When a password isn't required, EDB*Plus doesn't prompt for a password, such as when the `trust` authentication method is specified in the `pg_hba.conf` file. For more information about the `pg_hba.conf` file and authentication methods, see the PostgreSQL core documentation.

`connectstring` is the database connection string. See Using EDB*Plus for more information about the database connection string.

`variable` is a variable defined in the `login.sql` file that contains a database connection string. The `login.sql` file is in the `edbplus` subdirectory of the EDB Postgres Advanced Server home directory.

In this example, the database connection is changed to database `edb` on the localhost at port `5445` with username `smith`.

```
SQL> CONNECT smith/mypassword@localhost:5445/edb
Disconnected from EnterpriseDB Database.
Connected to EnterpriseDB 16.4.1 (localhost:5445/edb) AS smith
```

In this session, the connection is changed to the username `enterprisedb`. The host defaults to the localhost, the port defaults to `5444` (which isn't the same as the port previously used), and the database defaults to `edb`.

```
SQL> CONNECT enterprisedb/password
Disconnected from EnterpriseDB Database.
Connected to EnterpriseDB 16.4.1 (localhost:5444/edb) AS enterprisedb
```

This example shows connectivity for a multi-node cluster (one primary node and two secondary nodes) setup. The given multi-host `connectstring` syntax is used to establish a connection with the active primary database server. In this case, using `CONNECT` command, the connection is established with the primary database node on host `192.168.22.24` at port `5444`.

```
SQL> CONNECT enterprisedb/edb@192.168.22.24:5444,192.168.22.25:5445,192.168.22.26:5446/edb?
targetServerType=primary
Disconnected from EnterpriseDB Database.
Connected to EnterpriseDB 16.4.1 (192.168.22.24:5444/edb) AS
enterprisedb
```

DEFINE

The `DEFINE` command creates or replaces the value of a *user variable* (also called a *substitution variable*).

```
DEF[INE ] [ variable [ = text ]
]
```

If you give the `DEFINE` command without any parameters, all current variables and their values are displayed.

If `DEFINE variable` is given, only `variable` is displayed with its value.

`DEFINE variable = text` assigns `text` to `variable.text`, which you can optionally enclose in single or double quotation marks. Quotation marks must be used if `text` contains space characters.

This example defines two variables, `dept` and `name`:

```
SQL> DEFINE dept =
20
SQL> DEFINE name = 'John
Smith'
SQL>
DEFINE
DEFINE EDB =
"localhost:5445/edb"
DEFINE DEPT = "20"
DEFINE NAME = "John
Smith"
```

> **Note**
>
> The variable `EDB` is read from the `login.sql` file located in the `edbplus` subdirectory of the EDB Postgres Advanced Server home directory.

DEL

`DEL` deletes one or more lines from the SQL buffer.

```
DEL [ n | n m | n * | n L[AST ] | * | * n | * L[AST ] | L[AST
] ]
```

The parameters specify the lines to delete from the SQL buffer. Two parameters specify the start and end of a range of lines to delete. Giving the `DEL` command without parameters deletes the current line.

`n`

`n` is an integer representing the nth line.

`n m`

`n` and `m` are integers, where `m` is greater than `n` representing the nth through the mth lines.

`*`

Current line.

`LAST`

Last line.

This example deletes the fifth and sixth lines, which contain columns `sal` and `comm`, respectively, from the `SELECT` command in the SQL buffer.

```
SQL> LIST
  1  SELECT
  2    empno
  3
,ename
  4   ,job
  5   ,sal
  6   ,comm
  7   ,deptno
  8* FROM
emp
SQL> DEL 5
6
SQL> LIST
  1  SELECT
  2    empno
  3
,ename
  4   ,job
  5   ,deptno
  6* FROM
emp
```

DESCRIBE

The `DESCRIBE` command displays:

- A list of columns, column data types, and column lengths for a table or view
- A list of parameters for a procedure or function
- A list of procedures and functions and their respective parameters for a package

The `DESCRIBE` command also displays the structure of the database object referred to by a synonym. The syntax is:

```
DESC[RIBE] [
schema.]object
```

`schema`

Name of the schema containing the object to describe.

`object`

Name of the table, view, procedure, function, or package to display or the synonym of an object.

DISCONNECT

The `DISCONNECT` command closes the current database connection but doesn't end the EDB*Plus session.

```
DISC[ONNECT
]
```

EDIT

The `EDIT` command invokes an external editor to edit the contents of an operating system file or the SQL buffer.

```
ED[IT ] [ filename[.ext ]
]
```

`filename[.ext ]`

`filename` is the name of the file to open with an external editor. `ext` is the filename extension. If the extension is `sql`, then you can omit it. (`EDIT` assumes a `.sql` extension on filenames that are specified with no extension.) If you omit the `filename` parameter from the `EDIT` command, the contents of the SQL buffer are brought into the editor.

EXECUTE

The `EXECUTE` command executes an SPL procedure from EDB*Plus.

```
EXEC[UTE ] spl_procedure [ ([ parameters ])
]
```

`spl_procedure`

The name of the SPL procedure to execute.

`parameters`

Comma-delimited list of parameters. If there are no parameters, then you can optionally specify a pair of empty parentheses.

EXIT

The `EXIT` command ends the EDB*Plus session and returns control to the operating system. `QUIT` is a synonym for `EXIT`. Specifying no parameters is equivalent to `EXIT SUCCESS COMMIT`.

```
{ EXIT | QUIT
}
[ SUCCESS | FAILURE | WARNING | value | variable
]
[ COMMIT | ROLLBACK ]SUCCESS | FAILURE
|WARNING]
```

Returns an operating-system-dependent return code indicating successful operation, failure, or warning for `SUCCESS`, `FAILURE`, and `WARNING`, respectively. The default is `SUCCESS`.

`value`

An integer value returned as the return code.

`variable`

A variable created with the `DEFINE` command whose value is returned as the return code.

`COMMIT | ROLLBACK`

If you specify `COMMIT`, uncommitted updates are committed upon exit. If you specify `ROLLBACK`, uncommitted updates are rolled back upon exit. The default is `COMMIT`.

GET

The `GET` command loads the contents of the given file to the SQL buffer.

```
GET filename[.ext ] [ LIS[T ] | NOL[IST ]
]
```

`filename[.ext ]`

`filename` is the name of the file to load into the SQL buffer. `ext` is the filename extension. If the extension is `sql`, then you can omit it. `GET` assumes a `.sql` extension on filenames that are specified with no extension.

`LIST | NOLIST`

If you specify `LIST`, the contents of the SQL buffer are displayed after the file is loaded. If you specify `NOLIST`, no listing is displayed. The default is `LIST`.

HELP

The `HELP` command gets an index of topics or help on a specific topic. The question mark `(?)` is synonymous with specifying `HELP`.

```
{ HELP | ? } { INDEX | topic
}
```

`INDEX`

Displays an index of available topics.

`topic`

The name of a specific topic, such as an EDB*Plus command, for which you want help.

HOST

The `HOST` command executes an operating system command from EDB*Plus.

```
HO[ST ]
[os_command]
```

`os_command`

The operating system command to execute. If you don't provide an operating system command, EDB*Plus pauses and opens a new shell prompt. When the shell exits, EDB*Plus resumes execution.

INPUT

The `INPUT` command adds a line of text to the SQL buffer after the current line.

```
I[NPUT ]
text
```

This sequence of `INPUT` commands constructs a `SELECT` command:

```
SQL> INPUT SELECT empno, ename, job, sal,
comm
SQL> INPUT FROM
emp
SQL> INPUT WHERE deptno =
20
SQL> INPUT ORDER BY empno
SQL> LIST
  1 SELECT empno, ename, job, sal,
comm
  2 FROM
emp
  3 WHERE deptno =
20
  4* ORDER BY empno
```

LIST

`LIST` displays the contents of the SQL buffer.

```
L[IST] [ n | n m | n * | n L[AST] | * | * n | * L[AST] |
L[AST] ]
```

The buffer doesn't include a history of the EDB*Plus commands.

`n`

`n` represents the buffer line number.

`n m`

`n m` displays a list of lines between `n` and `m`.

`n *`

`n *` displays a list of lines that range between line `n` and the current line.

`n L[AST]`

`n L[AST]` displays a list of lines that range from line `n` through the last line in the buffer.

`*`

`*` displays the current line.

`* n`

`* n` displays a list of lines that range from the current line through line `n`.

`* L[AST]`

`* L[AST]` displays a list of lines that range from the current line through the last line.

`L[AST]`

`L[AST]` displays the last line.

PASSWORD

Use the `PASSWORD` command to change your database password.

```
PASSW[ORD]
[user_name]
```

You must have privileges to use the `PASSWORD` command to change another user's password. This example uses the `PASSWORD` command to change the password for a user named acctg:

```
SQL> PASSWORD acctg
Changing password for acctg
    New password:
    New password
again:
Password successfully
changed.
```

PAUSE

The `PAUSE` command displays a message and waits for the user to press `ENTER`.

```
PAU[SE]  [optional_text]
```

`optional_text` specifies the text to display to the user. If you omit `optional_text`, EDB Postgres Advanced Server displays two blank lines. If you double quote the `optional_text` string, the quotes are included in the output.

PROMPT

The `PROMPT` command displays a message to the user before continuing.

```
PRO[MPT]
[message_text]
```

`message_text` specifies the text displayed to the user. Double quote the string to include quotes in the output.

QUIT

The `QUIT` command ends the session and returns control to the operating system. `QUIT` is a synonym for `EXIT`.

```
QUIT

[SUCCESS | FAILURE | WARNING | value |
sub_variable]

[COMMIT |
ROLLBACK]
```

The default value is `QUIT SUCCESS COMMIT`.

REMARK

Use `REMARK` to include comments in a script.

```
REM[ARK] [optional_text]
```

You can also use the following convention to include a comment:

```
/*
 *  This is an example of a three-line comment.
 */
```

SAVE

Use the `SAVE` command to write the SQL buffer to an operating system file.

```
SAV[E] file_name
[CRE[ATE] | REP[LACE] |
APP[END]]
```

`file_name`

`file_name` specifies the name of the file (including the path) where the buffer contents are written. If you don't provide a file extension, `.sql` is appended to the end of the filename.

`CREATE`

Include the `CREATE` keyword to create a file. A file is created only if a file with the specified name doesn't already exist. This is the default.

`REPLACE`

Include the `REPLACE` keyword to overwrite an existing file.

`APPEND`

Include the `APPEND` keyword to append the contents of the SQL buffer to the end of the specified file.

This example saves the contents of the SQL buffer to a file named `example.sql`, located in the `temp` directory:

```
SQL> SAVE C:\example.sql
CREATE
File "example.sql"
written.
```

SET

Use the `SET` command to specify a value for a session-level variable that controls EDB*Plus behavior. The following forms of the `SET` command are valid.

SET AUTOCOMMIT`

Use the `SET AUTOCOMMIT` command to specify commit behavior for EDB Postgres Advanced Server transactions.

```
SET AUTO[COMMIT]

{ON | OFF | IMMEDIATE |
statement_count}
```

EDB*Plus always automatically commits DDL statements.

`ON`

Specify `ON` to turn on `AUTOCOMMIT` behavior.

`OFF`

Specify `OFF` to turn off `AUTOCOMMIT` behavior.

`IMMEDIATE`

`IMMEDIATE` has the same effect as `ON`.

`statement_count`

Include a value for `statement_count` to issue a commit after the specified count of successful SQL statements.

SET COLUMN SEPARATOR

Use the `SET COLUMN SEPARATOR` command to specify the text to display between columns.

```
SET COLSEP
column_separator
```

The default value of `column_separator` is a single space.

SET ECHO

Use the `SET ECHO` command to specify to display SQL and EDB*Plus script statements onscreen as they execute.

```
SET ECHO {ON |
OFF}
```

The default value is `OFF`.

SET FEEDBACK

The `SET FEEDBACK` command controls the display of interactive information after a SQL statement executes.

```
SET FEED[BACK] {ON | OFF |
row_threshold}
```

`row_threshold`

Specify an integer value for `row_threshold`. Setting `row_threshold` to `0` is same as setting `FEEDBACK` to `OFF`. Setting `row_threshold` equal `1` effectively sets `FEEDBACK` to `ON`.

SET FLUSH

Use the `SET FLUSH` command to control display buffering.

```
SET FLU[SH] {ON |
OFF}
```

Set `FLUSH` to `OFF` to enable display buffering. If you enable buffering, messages bound for the screen might not appear until the script completes. Setting `FLUSH` to `OFF` offers better performance.

Set `FLUSH` to `ON` to disable display buffering. If you disable buffering, messages bound for the screen appear immediately.

SET HEADING

Use the `SET HEADING` variable to specify whether to display column headings for `SELECT` statements.

```
SET HEA[DING] {ON |
OFF}
```

SET HEADSEP

The `SET HEADSEP` command sets the new heading separator character used by the `COLUMN HEADING` command. The default is `'|'`.

```
SET HEADS[EP]
```

SET LINESIZE

Use the `SET LINESIZE` command to specify the width of a line in characters.

```
SET LIN[ESIZE] width_of_line
```

`width_of_line`

The default value of `width_of_line` is `132`.

SET NEWPAGE

Use the `SET NEWPAGE` command to specify how many blank lines to print after a page break.

```
SET NEWP[AGE]
lines_per_page
```

`lines_per_page`

The default value of `lines_per_page` is `1`.

SET NULL

Use the `SET NULL` command to specify a string to display when a `NULL` column value is displayed in the output buffer.

```
SET NULL
null_string
```

SET PAGESIZE

Use the `SET PAGESIZE` command to specify the number of printed lines that fit on a page.

```
SET PAGES[IZE] line_count
```

Use the `line_count` parameter to specify the number of lines per page.

SET SQLCASE

The `SET SQLCASE` command specifies whether to convert SQL statements transmitted to the server to upper or lower case.

```
SET SQLC[ASE] {MIX[ED] | UP[PER] |
LO[WER]}
```

`UPPER`

Specify `UPPER` to convert the command text to upper case.

`LOWER`

Specify `LOWER` to convert the command text to lower case.

`MIXED`

Specify `MIXED` to leave the case of SQL commands unchanged. The default is `MIXED`.

SET PAUSE

The `SET PAUSE` command is most useful when included in a script. The command displays a prompt and waits for the user to press **Return**.

```
SET PAU[SE] {ON |
OFF}
```

If `SET PAUSE` is `ON`, the message `Hit ENTER to continue…` appears before each command executes.

SET SPACE

Use the `SET SPACE` command to specify the number of spaces to display between columns.

```
SET SPACE
number_of_spaces
```

SET SQLPROMPT

Use `SET SQLPROMPT` to set a value for a user-interactive prompt.

```
SET SQLP[ROMPT]
"prompt"
```

By default, `SQLPROMPT` is set to `"SQL> "`

SET TERMOUT

Use the `SET TERMOUT` command to specify to display command output.

```
SET TERM[OUT] {ON |
OFF}
```

SET TIMING`

The `SET TIMING` command specifies whether to display the execution time for each SQL statement after it executes.

```
SET TIMI[NG] {ON |
OFF}
```

SET TRIMSPOOL`

Use the `SET TRIMSPOOL` command to remove trailing spaces from each line in the output file specified by the `SPOOL` command.

```
SET TRIMS[POOL] {ON |
OFF}
```

The default value is `OFF`.

SET VERIFY

Specifies whether to display both the old and new values of a SQL statement when a substitution variable is encountered.

```
SET VER[IFY] { ON | OFF
}
```

SHOW

Use the `SHOW` command to display current parameter values.

```
SHO[W] {ALL |
parameter_name}
```

Display the current parameter settings by including the `ALL` keyword:

```
SQL> SHOW ALL
autocommit      OFF
colsep          "
"
define          "&"
echo            OFF
FEEDBACK ON for 6 row(s).
flush           ON
heading         ON
headsep         "|"
linesize        78
newpage         1
null            "
"
pagesize        14
pause           OFF
serveroutput    OFF
spool           OFF
sqlcase         MIXED
sqlprompt       "SQL>
"
sqlterminator   ";"
suffix          ".sql"
termout         ON
timing          OFF
verify          ON
USER is         "enterprisedb"
HOST is         "localhost"
PORT is         "5444"
DATABASE is     "edb"
VERSION is      "14.0.0"
```

Or display a specific parameter setting by including the `parameter_name` in the `SHOW` command:

```
SQL> SHOW VERSION
VERSION is "14.0.0"
```

SPOOL

The `SPOOL` command sends output from the display to a file.

```
SP[OOL] output_file |
OFF
```

Use the `output_file` parameter to specify a pathname for the output file.

START

Use the `START` command to run an EDB*Plus script file. `START` is an alias for the `@` command.

```
STA[RT]
script_file
```

Specify the name of a script file in the `script_file` parameter.

UNDEFINE

The `UNDEFINE` command erases a user variable created by the `DEFINE` command.

```
UNDEF[INE] variable_name [
variable_name...]
```

Use the `variable_name` parameter to specify the name of a variable or variables.

WHENEVER SQLERROR

The `WHENEVER SQLERROR` command provides error handling for SQL errors or PL/SQL block errors. The syntax is:

```
WHENEVER
SQLERROR

{CONTINUE[COMMIT|ROLLBACK|NONE]
  |EXIT[SUCCESS|FAILURE|WARNING|n|sub_variable]

[COMMIT|ROLLBACK]}
```

If EDB Postgres Advanced Server encounters an error while executing a SQL command or PL/SQL block, EDB*Plus performs the action specified in the `WHENEVER SQLERROR` command:

- Include the `CONTINUE` clause to perform the specified action before continuing.
- Include the `COMMIT` clause to commit the current transaction before exiting or continuing.
- Include the `ROLLBACK` clause to roll back the current transaction before exiting or continuing.
- Include the `NONE` clause to continue without committing or rolling back the transaction.
- Include the `EXIT` clause to perform the specified action and exit in case of an error.

Use the following options to specify a status code that EDB*Plus returns before exiting:

```
[SUCCESS|FAILURE|WARNING|n|sub_variable]
```

EDB*Plus supports substitution variables but doesn't support bind variables.