



EDB*Plus User's Guide

リリース *14.1.0*

EnterpriseDB

2021 年 12 月 16 日

目次

第 1 章 EDB*Plus	3
第 2 章 Installing EDB*Plus	5
2.1 インストールの前提条件	5
2.2 CentOS ホストへの EDB * Plus のインストール	6
2.3 EDB * Plus を RHEL ホストにインストールする	7
2.4 CentOS / RHEL 7 ppc64le ホストへの EDB * Plus のインストール	8
2.5 Debian または Ubuntu ホストへの EDB * Plus のインストール	10
2.6 RPM インストールの構成	11
2.7 グラフィカルインストーラーの使用	12
第 3 章 Using EDB*Plus	17
第 4 章 Using a Secure Sockets Layer (SSL) Connection	21
4.1 Advanced Server で SSL を構成する	22
4.2 EDB * Plus クライアント用の SSL の構成	24
4.3 EDB * Plus と Advanced Server データベース間の SSL 接続のリクエスト	26
第 5 章 Command Summary	29
5.1 受け入れる	29
5.2 追加	29
5.3 変更	30
5.4 クリア	30
5.5 列	31
5.6 接続	34
5.7 定義	35
5.8 DEL	35
5.9 説明	36
5.10 切断	37
5.11 編集	37
5.12 実行	37
5.13 終了	38
5.14 GET	38
5.15 ヘルプ	39
5.16 ホスト	39
5.17 入力	39
5.18 リスト	40
5.19 パスワード	40
5.20 一時停止	41
5.21 プロンプト	41
5.22 終了	41

5.23	備考	42
5.24	保存	42
5.25	セット	43
5.26	ショー	46
5.27	スプール	47
5.28	開始	47
5.29	未定義	48
5.30	いつでも <code>SQLERROR</code>	48

このガイドでは、EDB * Plus を使用して Advanced Server データベースに接続する方法について説明します。EDB * Plus は、次のことを可能にする SQL コマンドを受け入れる EDB Postgres Advanced Server へのコマンドラインユーザインタフェースを提供します。

- 特定のデータベースオブジェクトのクエリ
- ストアドプロシージャの実行 SQL コマンドからの出力のフォーマット
- バッチスクリプトの実行
- OS コマンドの実行
- レコード出力

Advanced Server がサポートする機能の詳細については、次の Web サイトで入手可能な Advanced Server ガイドの完全なライブラリを参照してください。

<https://www.enterprisedb.com/docs>

第1章 EDB*Plus

</ div>

EDB * Plus は、EDB Postgres Advanced Server へのコマンドラインユーザインタフェースを提供するユーティリティプログラムです。EDB * Plus は、SQL コマンド、SPL 無名ブロック、および EDB * Plus コマンドを受け入れます。

EDB * Plus コマンドはオラクル SQL * Plus コマンドと互換性があり、次のようなさまざまな機能を提供します。

- 特定のデータベースオブジェクトのクエリ
- ストアドプロシージャの実行 SQL コマンドからの出力のフォーマット
- バッチスクリプトの実行
- OS コマンドの実行
- 記録出力

第2章 Installing EDB*Plus

</div>

RPM インストーラまたはグラフィカルインストーラを使用して、EDB *Plus を Advanced Server インストールに追加できます。

2.1 インストールの前提条件

EDB *Plus をインストールする前に、最初に Java（バージョン 1.8 以降）をインストールする必要があります。Linux システムでは、yum パッケージマネージャを使用して Java をインストールできます。ターミナルウィンドウを開き、スーパーユーザ権限を引き受けて、次のように入力します。

- RHEL または CentOS 7 の場合：

```
* # yum -y install java
```

- RHEL または CentOS 8 の場合：

```
* # dnf -y install java
```

Windows を使用している場合、Java インストーラーと手順はオンラインで利用できます：

< [http://www.java.com/en/ ダウンロード/ マニュアル.jsp](http://www.java.com/en/ダウンロード/マニュアル.jsp) >

EDB リポジトリへのアクセスを許可する資格情報も必要です。資格情報のリクエストについては、次をご覧ください。

< <https://www.enterprisedb.com/ ユーザ/ ログイン> >

リポジトリの資格情報を受け取った後：

1. リポジトリ設定ファイルを作成します。ファイルを変更し、ユーザ名前とパスワードを入力します。EDB *Plus をインストールします。

EDB リポジトリの作成および使用による Advanced Server またはそのサポートコンポーネントのインストールの詳細については、次の Web サイトで入手可能な* EDB Postgres Advanced Server インストールガイド* を参照してください。

<https://www.enterprisedb.com/docs>

2.2 CentOS ホストへの EDB * Plus のインストール

RPM パッケージを使用して、CentOS ホストに EDB * Plus をインストールできます。

- リポジトリ構成ファイルをインストールするには、スーパーユーザ特権を想定し、次のプラットフォーム固有のコマンドのいずれか呼び出します。

– CentOS 7 の場合：

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

- CentOS 8 の場合：

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

- 次のコマンドの USERNAME:PASSWORD 変数を、登録済みの EDB ユーザのユーザー名とパスワードに置き換えます。

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

EDB * Plus をインストールする前に、epel-release パッケージをインストールする必要があります。

- CentOS 7 の場合：

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

- CentOS 8 の場合：

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

- CentOS 8 の場合、PowerTools リポジトリを有効にしてパッケージの依存関係を満たします。

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

- リポジトリ構成ファイルの名前付けは edb.repo です。ファイルは/etc/yum.repos.d にあります。構成ファイルへの変更を保存した後、次のコマンドを使用して EDB * Plus をインストールできます。

– CentOS 7 の場合：

```
* * * yum -y install edb-asxx-edbplus
```

- CentOS 8 の場合：

```
* * dnf -y install edb-asxx-edbplus
```

ここで、**xx** は Advanced Server バージョンです。

システムで認識されないソースによって署名された RPM パッケージをインストールすると、**yum** がローカルサーバにキーをインポートするパーミッションを求める場合があります。プロンプトが表示され、パッケージソフトが信頼できるソースからのものであることに満足したら、**y** と入力し、**Return** を押して続行します。

インストール中に、**yum** が解決できない依存関係に遭遇する場合があります。存在する場合、手動で解決する必要がある依存関係のリストが提供されます。

2.3 EDB * Plus を RHEL ホストにインストールする

RPM パッケージを使用して、RHEL ホストに EDB * Plus をインストールできます。

- リポジトリ構成ファイルをインストールするには、スーパーユーザ特権を想定し、次のプラットフォーム固有のコマンドのいずれかを呼び出します。

- RHEL 7 の場合：

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

- RHEL 8 の場合：

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

- 次のコマンドの **USERNAME:PASSWORD** 変数を、登録済みの EDB ユーザのユーザー名とパスワードに置き換えます。

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

EDB * Plus をインストールする前に、**epel-release** パッケージをインストールする必要があります。

- RHEL 7 の場合：

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

- RHEL 8 の場合：

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

- リポジトリを有効にします。
 - RHEL7 で、**optional**、**extras** および **HA** リポジトリを有効にして、パッケージの依存関係を満たします。

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

- RHEL 8 では、`codeready-builder-for-rhel-8-*-rpms` リポジトリを有効にしてパッケージの依存関係を満たします。

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

- リポジトリ構成ファイルの名前付けは `edb.repo` です。ファイルは `/etc/yum.repos.d` にあります。構成ファイルへの変更を保存した後、次のコマンドを使用して EDB *Plus をインストールできます。

– RHEL 7 の場合：

```
* * * yum -y install edb-asxx-edbplus
```

- RHEL 8 の場合：

```
* * dnf -y install edb-asxx-edbplus
```

ここで、**xx** は Advanced Server バージョンです。

システムで認識されないソースによって署名された RPM パッケージをインストールすると、`yum` がローカルサーバにキーをインポートするパーミッションを求める場合があります。プロンプトが表示され、パッケージソフトが信頼できるソースからのものであることに満足したら、`y` と入力し、`Return` を押して続行します。

インストール中に、`yum` が解決できない依存関係に遭遇する場合があります。存在する場合、手動で解決する必要がある依存関係のリストが提供されます。

2.4 CentOS / RHEL 7 ppc64le ホストへの EDB *Plus のインストール

RPM パッケージを使用して、CentOS または RHEL 7 ppc64le ホストに EDB *Plus をインストールできます。

IBM Advance Toolchain リポジトリをインストールするには：

- CentOS または RHEL 7 ppc64le の場合：

```
* rpm --import https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/  
→redhat/RHEL7/gpg-pubkey-6976a827-5164221b
```

リポジトリ構成ファイルの名前付けは `advance-toolchain.repo` です。ファイルは `/etc/yum.repos.d` にあります。

- `advance-toolchain.repo` ファイルの作成後、デフォルトで `enabled` パラメータが 1 に設定されデフォルト。

```
* [advance-toolchain]
* name=Advance Toolchain IBM FTP
* baseurl=https://public.dhe.ibm.com/software/server/POWER/Linux/
* toolchain/at/redhat/RHEL7
* failovermethod=priority
* enabled=1
* gpgcheck=1
* gpgkey=ftp://public.dhe.ibm.com/software/server/POWER/Linux/
* toolchain/at/redhat/RHELX/gpg-pubkey-6976a827-5164221b
```

EDB リポジトリ構成ファイルをインストールするには、スーパーユーザ権限を引き受け、次のコマンドを呼び出します。

- CentOS または RHEL 7 ppc64le の場合：

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

- 次のコマンドの USERNAME:PASSWORD 変数を、登録済みの EDB ユーザのユーザー名とパスワードに置き換えます。

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

EDB *Plus をインストールする前に、`epel-release` パッケージをインストールする必要があります。

- CentOS または RHEL 7 ppc64le の場合：

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

- リポジトリを有効にします。
 - RHEL7 で、`optional`、`extras` および `HA` リポジトリを有効にして、パッケージの依存関係を満たします。

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

- リポジトリ構成ファイルの名前付けは `edb.repo` です。ファイルは `/etc/yum.repos.d` にあります。構成ファイルへの変更を保存した後、次のコマンドを使用して EDB *Plus をインストールできます。

```
* * * yum -y install edb-asxx-edbplus
```

ここで、`xx` は Advanced Server バージョンです。

システムで認識されないソースによって署名された RPM パッケージをインストールすると、`yum` がローカルサーバにキーをインポートするパーミッションを求める場合があります。プロンプトが表示され、パッケージソフトが信頼できるソースからのものであることに満足したら、`y` と入力し、`Return` を押して続行します。

インストール中に、yum が解決できない依存関係に遭遇する場合があります。存在する場合、手動で解決する必要がある依存関係のリストが提供されます。

2.5 Debian または Ubuntu ホストへの EDB * Plus のインストール

Debian または Ubuntu ホストに EDB * Plus をインストールするには、EDB リポジトリへのアクセスを許可する資格情報が必要です。リポジトリの資格情報を要求するには、次をご覧ください。

<https://www.enterprisedb.com/repository-access-request>

次の手順では、EDB apt リポジトリを使用して debian パッケージをインストールします。コマンドを使用する場合、username および password を EDB が提供する資格情報に置き換えます。

- スーパーユーザの権限を想定：

```
* sudo su -
```

EDB リポジトリを構成します。

- Debian 9、Ubuntu 18、および Ubuntu 20：

```
* sh -c 'echo "deb https://USERNAME:PASSWORD@apt.enterprisedb.com/$(lsb_release -
↪cs)-edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -cs).
↪list'
```

- Debian 10 の場合：
- a。 EDB リポジトリをセットアップします。

```
* sh -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'
```

- b。 次のコマンドで、username および password プレースホルダーを EDB 資格情報に置き換えます。

```
* sh -c 'echo "machine apt.enterprisedb.com login <USERNAME> password <PASSWORD>" >
↪/etc/apt/auth.conf.d/edb.conf'
```

- 安全な APT リポジトリのサポートをシステムに追加します。

```
* apt-get -y install apt-transport-https
```

EDB 署名キーを追加します。

```
* wget -q -O - https://apt.enterprisedb.com/edb-deb.gpg.key | sudo apt-key add -
```

- リポジトリメタデータの更新：

```
* apt-get update
```

- Debian パッケージのインストール：

```
* apt-get -y install edb-asxx-edbplus
```

ここで、**xx** は Advanced Server バージョンです。

2.6 RPM インストールの構成

EDB * Plus の RPM インストールを実行した後、EDB * Plus が Java インストールを見つけることができる環境変数の値を設定する必要があります。次のコマンドを使用して、変数数値を設定します。

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

デフォルトでは、RPM インストーラの `pg_hba.conf` ファイルは `IDENT` 認証を強制します。EDB * Plus を呼び出す前に、`pg_hba.conf` ファイルを変更し、認証メソッドを `IDENT` 以外のフォームに変更する（およびサーバーを再起動する）か、次の手順を実行して保証サーバーにアクセスできるようにする必要があります。

`identd` サーバーがインストールされ、実行されていることを確認する必要があります。次のコマンドを呼び出して、`yum` パッケージマネージャを使用して `identd` サーバーをインストールできます。

- RHEL または CentOS 7 の場合：

```
* yum -y install xinetd authd
```

- RHEL または CentOS 8 の場合：

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

コマンドは、次を含む `/etc/xinetd.d/auth` という名前付けのファイルを作成する必要があります。

```
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 * ファイルのサーバー引数の最後に `-E` 引数が含まれている場合は、`-E` を消去してください。

次に、`identd` サーバーをスタートするには、次のコマンドを呼び出します。

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

pg_ident.conf ファイルを開き、ユーザマッピングを作成します。

#	map_name	system_username	postgres_username
*	edbas	enterprisedb	enterprisedb

どこで：

- map_name 列に指定された名前は、pg_hba.conf ファイル内のマッピングを識別するユーザー定義の名前です。
- system_username 列に指定されている名前は enterprisedb です。
- postgres_username 列に指定されている名前は enterprisedb です。

次に、pg_hba.conf ファイルを開き、IDENT エントリを変更します。

- IPv4 ローカル接続を使用している場合は、ファイルエントリを次のように変更します。


```
- host all all 127.0.0.0/0 ident map=edbas
```
- IPv6 ローカル接続を使用している場合、ファイルエントリを次のように変更します。


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

EDB * Plus を呼び出す前に、Advanced Server サービスをリスタートする必要があります。Advanced Server サービスの制御に関する詳細については、次から入手できる* EDB Postgres Advanced Server インストールガイド*を参照してください。

<https://www.enterprisedb.com/docs>

2.7 グラフィカルインストーラーの使用

EDB * Plus のグラフィカルインストーラーは、StackBuilder Plus から入手できます。Windows または Linux のスタートメニューから StackBuilder Plus にアクセスできます。StackBuilder Plus を開き、EDB * Plus をインストールするインストールを選択した後、コンポーネント選択スクリーンのツリーコントロールを展開して、EDB * Plus インストーラを選択およびダウンロードします。

図 1：EDB* Plus のようこそウィンドウ

EDB * Plus インストーラは、次の図に示すように、セットアップウィザードによるこそ。

図 2：[インストールディレクトリ] ウィンドウ

Installation Directory フィールドを使用して、EDB * Plus ソフトウェアをインストールするディレクトリを指定します。次に、Next をクリックして続行します。

図 3：[Advanced Server Installation Details] ウィンドウ

EDB Postgres Advanced Server Installation Details ウィンドウのフィールドを使用して、Advanced Server ホストの場所を特定します。

- Host フィールドを使用して、Advanced Server が存在するシステムを識別します。
- Port フィールドを使用して、Advanced Server がクライアント接続を監視するリスナポートを識別します。

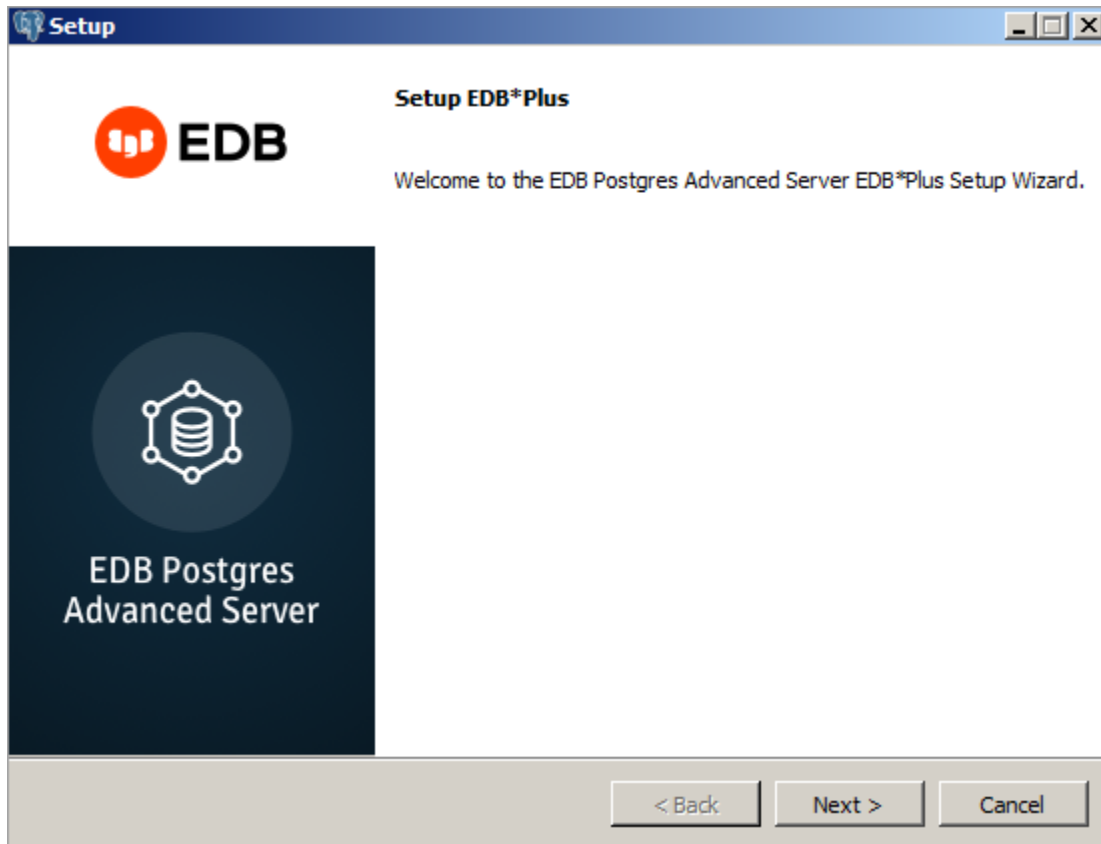


図 1: The EDBPlus Welcome window

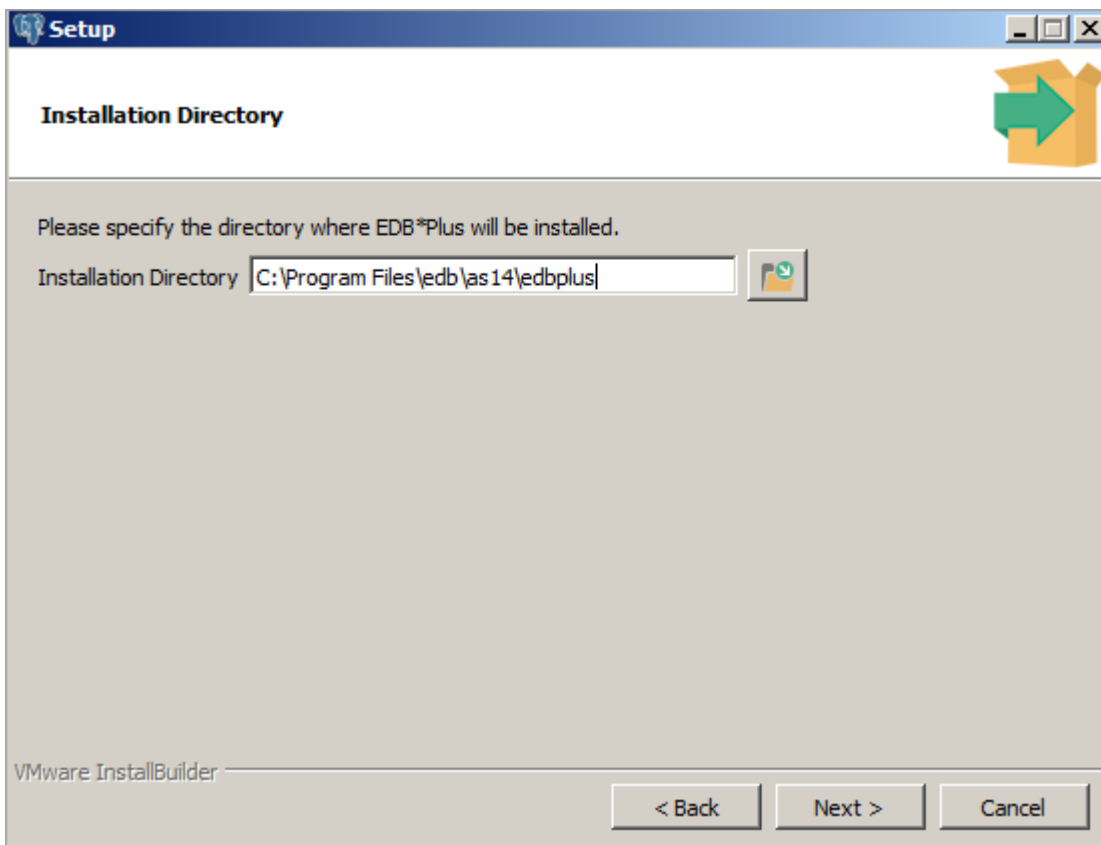


図 2: The Installation Directory window

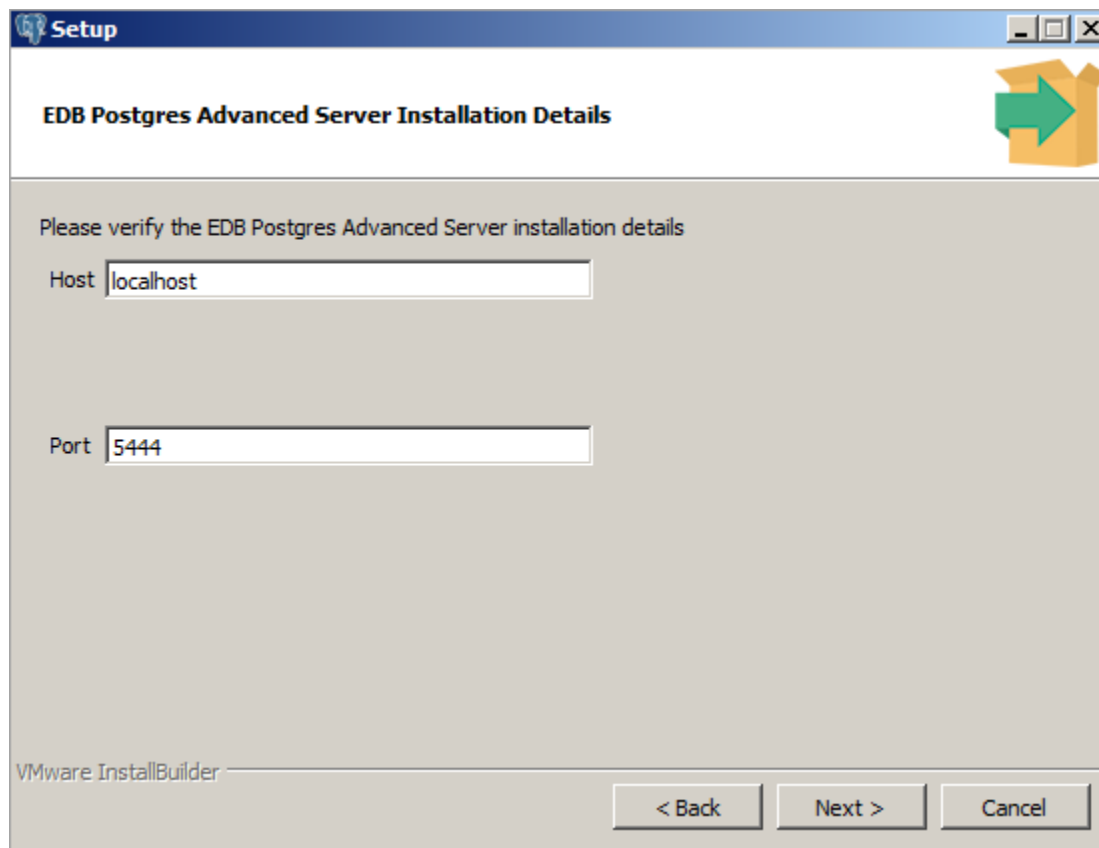


図 3: The Advanced Server Installation Details window

次に、**Next** をクリックして続行します。

図 4：インストールの準備完了ウィンドウ

Ready to Install ウィンドウは、システムに EDB * Plus をインストールするために必要な情報がすべてインストーラにある場合に通知します。**Next** をクリックして、EDB * Plus をインストールします。

図 5：インストールが完了しました

セットアップウィザードが EDB * Plus のインストールを完了すると、インストーラから通知されます。**Finish** をクリックして、インストーラを終了します。

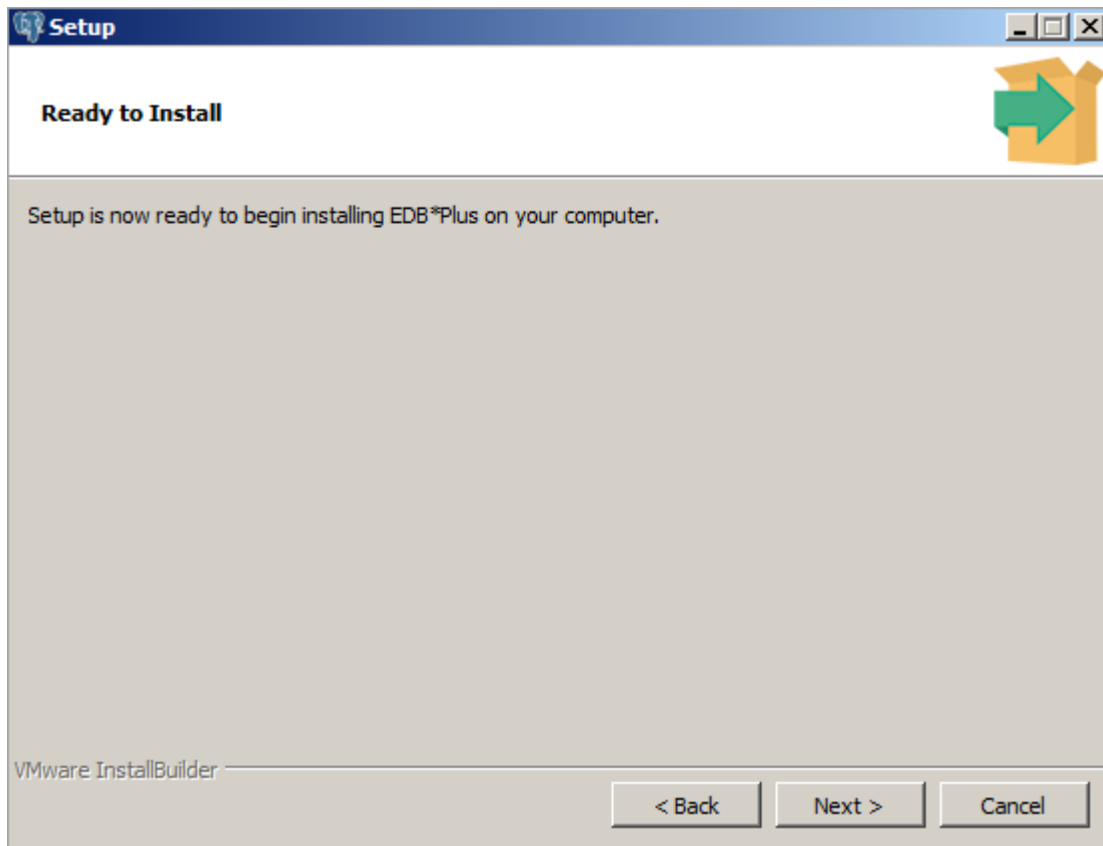


図 4: The Ready to Install window

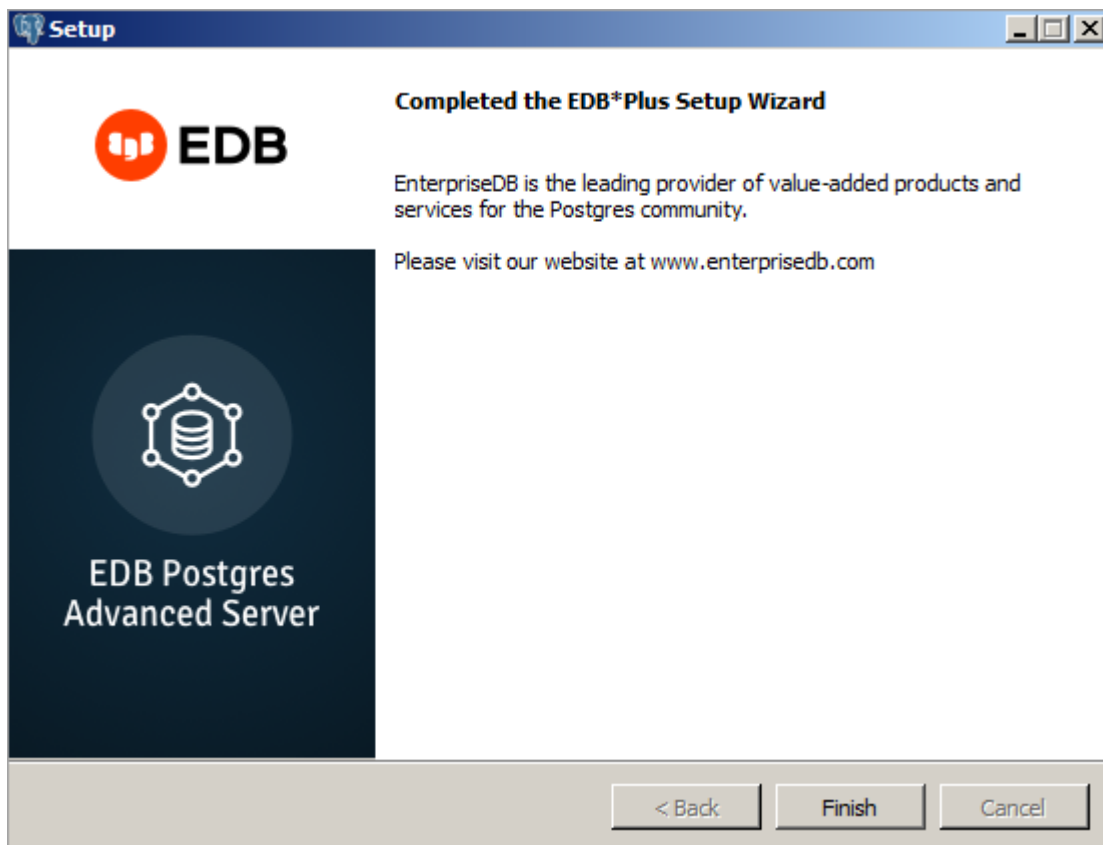


図 5: The installation is complete

第3章 Using EDB*Plus

</div>

</div>

EDB *Plus コマンドラインを開くには、Applications または Start メニューから Advanced Server メニュー、Run SQL Command Line メニューに移動し、EDB *Plus オプションを選択します。次のコマンドを使用して、オペレーティングシステムのコマンドラインから EDB *Plus を呼び出すこともできます。

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

SILENT

指定した場合、EDB *Plus サインオンバナーはすべてのプロンプトとともに抑制されます。

login

データベースサーバとデータベースに接続するためのログイン情報。login は次のフォームを取ります。ログイン情報内に空白文字があってはなりません。

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

どこで：

username は、データベースに接続するためのデータベースユーザー名です。

password は、指定された username に関連付けられたパスワード。password が提供されないが、認証にパスワードが必要な場合、パスワードファイルが使用可能であれば使用されます。パスワードファイルがないか、マッチング接続パラメーターを持つパスワードファイルにエントリーがない場合、EDB *Plus はパスワードのプロンプトを求めます。

connectstring は、次のフォーマットのデータベースコネクション文字列です。

```
<host>[:<port>][/<dbname>][?ssl={true | false}]
```

どこで：

host は、データベースサーバが存在するホスト名または IP アドレスです。@connectstring も @variable も /NOLOG も指定されていない場合、デフォルトのホストはローカルホストであると想定されます。port は、データベースサーバで接続を受信するポート番号です。指定しない場合、デフォルトは 5444 です。dbname は、接続するデータベースの名前です。指定しない場合、デフォルトは edb です。接続に IPv4 ではなく Internet Protocol version 6 (IPv6) を使用する場合は、IP アドレスを大括弧で囲む必要があります（つまり、[ipv6_address]）。以下は、IPv6 接続を使用した例です。

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

データベースサーバの `pg_hba.conf` ファイルには、IPv6 接続の適切なエントリーが含まれている必要があります。次の例は、すべてのアドレスを許可するエントリーを示しています。

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

`pg_hba.conf` ファイルの詳細については、次の PostgreSQL コア文書を参照してください。

< [https://www.postgresql.org/docs/current/ 静的/auth-pg-hba-conf.html](https://www.postgresql.org/docs/current/静的/auth-pg-hba-conf.html) >

SSL 接続が必要な場合は、接続文字列に `?ssl=true` パラメータを含めます。そのような場合、接続文字列には、`/dbname` の有無にかかわらず、最小限、`host:port` を含める必要があります。`ssl` パラメータが指定されていない場合、デフォルトは `false` です。SSL 接続のセットアップ手順については、[Using a Secure Sockets Layer \(SSL\) Connection](#) を参照してください。

`variable` は、データベースコネクション文字列を含む `login.sql` ファイルで定義された変数。`login.sql` ファイルは、Advanced Server ホームディレクトリサブディレクトリにあります。

`/NOLOG`

データベースコネクションを確立せずに EDB * Plus をスタートするには、`/NOLOG` を指定します。データベースコネクションを必要とする SQL コマンドおよび EDB * Plus コマンドは、このモードでは使用できません。`/NOLOG` オプションを指定して EDB * Plus を起動した後、`CONNECT` コマンドを続けてデータベースに接続できます。

`scriptfile[.ext ]`

`scriptfile` は、カレントワーキングディレクトリにあるファイルの名前です EDB * Plus のスタートアップ後に自動的に実行される SQL および/または EDB * Plus コマンドが含まれています。`ext` はファイル名拡張子です。ファイル名の拡張子が `sql` の場合、`scriptfile` を指定するときに `.sql` 拡張子を省略できます。スクリプトファイルを作成するときは、必ず拡張子を付けてファイルに名前を付けてください。そうしないと、EDB * Plus からアクセスできなくなります。（EDB * Plus は、拡張子なしで指定されたファイル名で常に `.sql` 拡張子を想定します。）

次の例は、ポート 5444 の `localhost` 上のデータベースサーバで実行されているデータベース `edb` に接続する、パスワード `password` のユーザ `enterprisedb` を示しています。

```
C:\Program Files\edb\as14\edbplus>edbplus enterprisedb/password
Connected to EnterpriseDB 14.0.0 (localhost:5444/edb) AS enterprisedb

EDB*Plus: Release 14 (Build 40.0.0)
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.

SQL>
```

次の例では、パスワード `password` を持つユーザ `enterprisedb` が、ポート 5445 の `localhost` 上のデータベースサーバで実行されているデータベース `edb` に接続しています。

```
C:\Program Files\edb\as14\edbplus>edbplus enterprisedb/password@localhost:5445/edb
Connected to EnterpriseDB 14.0.0 (localhost:5445/edb) AS enterprisedb
```

```
EDB*Plus: Release 14 (Build 40.0.0)
```

```
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.
```

```
SQL>
```

login.sql ファイルの変数 hr_5445 を使用して、ポート 5445 のローカルホスト上のデータベース hr への接続にどのように使用されるかを以下に示します。

```
C:\Program Files\edb\as14\edbplus>edbplus enterprisedb/password@hr_5445
Connected to EnterpriseDB 14.0.0 (localhost:5445/hr) AS enterprisedb
```

```
EDB*Plus: Release 14 (Build 40.0.0)
```

```
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.
```

```
SQL>
```

以下は、前の例で使用した login.sql ファイルのコンテンツです。

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

次の例では、ポート 5444 でサーバーローカルホストのデータベース edb に接続した後、スクリプトファイル dept_query.sql を実行します。

```
C:\Program Files\edb\as14\edbplus>edbplus enterprisedb/password @dept_query
Connected to EnterpriseDB 14.0.0 (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.
```

以下は、前の例で使用されたファイル dept_query.sql のコンテンツです。

```
SET PAGESIZE 9999
SET ECHO ON
```

(次のページに続く)

(前のページからの続き)

```
SELECT * FROM dept;  
EXIT
```

第4章 Using a Secure Sockets Layer (SSL) Connection

</ div>

</ div>

Advanced Server データベースへの EDB * Plus 接続は、セキュアソケットレイア（SSL）接続を使用し
て実現できます。

SSL を使用するには、SSL 接続に関連するデータベースサーバで実行されるさまざまな前提条件の構成手
順と、EDB * Plus を実行するホストでの Java トラストストアおよびキーストアの作成が必要です。

Java * truststore * は、Java クライアント（EDB * Plus）が SSL 接続を開始するサーバーの信頼性を検証
するために使用する認証局（CA）証明書を含むファイル。

Java * keystore * は、プライベート鍵と公開鍵、およびそれらに対応する証明書を含むファイル。キースト
アは、EDB * Plus 接続に使用されるサーバーへのクライアント認証に必要です。

以下は、SSL 接続をセットアップする際のガイダンスとして参照できる資料です。

Advanced Server データベースへの SSL 接続のセットアップについては、次の場所にある PostgreSQL Core
ドキュメントの第 18 章「サーバーのセットアップと操作」の SSL を使用したセキュア TCP 接続に関する
セクションを参照してください。

- <https://www.postgresql.org/docs/current/ssl-tcp.html>
- SSL を使用した JDBC クライアント接続の詳細については、次の場所にある PostgreSQL JDBC イン
ターフェイスの第 4 章「SSL の使用」のクライアントの設定に関するセクションを参照してください。
- < [https://jdbc.postgresql.org/ 文書/head/ssl-client.html](https://jdbc.postgresql.org/文書/head/ssl-client.html) >

以下のセクションでは、SSL を使用する構成手順について説明します。

Advanced Server での SSL の構成 EDB * Plus クライアントの SSL の構成 - Advanced Advanced Server デー
タベースへの SSL 接続の要求

4.1 Advanced Server で SSL を構成する

このセクションでは、EDB * Plus で SSL を使用方法を示すために、データベースサーバで SSL を構成する例を示します。この目的には、自己署名証明書が使用されます。

ステップ 1: 証明書署名要求 (CSR) を作成します。

次の例では、生成された証明書署名要求ファイルは `server.csr` です。プライベートキーはファイル `server.key` として生成されます。

```
$ 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 * 証明書を作成するとき、コモンネームフィールドに指定された値（この例では `CN=enterprisedb` として指定）は、EDB * Plus への接続時に指定されたデータベースユーザ名前でなければなりません。

また、`pg_ident.conf` ファイルで定義されているユーザ名マップを使用して、コモンネームとデータベースユーザ名前の柔軟性を高めることができます。ステップ 8 および 9 では、ユーザ名前 `maps` の使用について説明します。

ステップ 2: 自己署名証明書を生成します。

以下は、証明書署名要求ファイル `server.csr` とプライベートキー `server.key` を入力として使用して、ファイル `server.crt` に自己署名証明書を生成します。

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

ステップ 3: サーバー証明書 (`server.crt`) のコピーを作成して、ルート認証局 (CA) ファイル (`root.crt`) として使用します。

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

ステップ 4: 冗長な証明書署名要求 (`server.csr`) を削除します。

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

ステップ 5: 証明書とプライベートキーファイルを Advanced Server データディレクトリ（例、`/opt/edb/as14/data`）に移動またはコピーします。

```
$ mv root.crt /opt/edb/as14/data
$ mv server.crt /opt/edb/as14/data
$ mv server.key /opt/edb/as14/data
```

ステップ 6: 証明書ファイルとプライベートキーファイルのファイル所有権と許可を設定します。

所有権を、データベースサーバのデータサブディレクトリを所有するオペレーティングシステムアカウントに設定します。所有者以外の他のグループまたはアカウントがこれらのファイルにアクセスできないように、アクセス許可を設定します。

```
$ 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
```

ステップ 7: postgresql.conf ファイルで、次の変更を make ます。

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

ステップ 8: pg_hba.conf ファイルを変更して、EDB * Plus が SSL 接続を make 対象のデータベースで SSL の使用を有効にします。

pg_hba.conf ファイルの hostssl タイプは、エントリーがクライアントからの SSL 接続試行を検証するために使用されることを示します (EDB * Plus)。

認証メソッドは、証明書のコモンネーム（この例では enterprisedb）を使用して認証が実行されるクライアントからの SSL 証明書を要求するオーダーに、オプション clientcert=1 で cert に設定されます。

map=sslusers オプションは、pg_ident.conf ファイルで定義された sslusers 名前付けのマッピングが認証に使用されることを指定します。このマッピングにより、証明書からのコモンネームと接続を試行するデータベースデータベースユーザ名前が pg_ident.conf ファイルにリストされている SYSTEM-USERNAME/PG-USERNAME ペアとマッチする場合、データベースへの接続が許可されます。

以下は、データベース (edb) が SSL 接続を使用する必要がある場合の pg_hba.conf ファイルの設定の例です。

#	TYPE	DATABASE	USER	ADDRESS	METHOD
#	"local" is for Unix domain socket connections only				

(次のページに続く)

(前のページからの続き)

```
local    all                all                                md5
# IPv4 local connections:
hostssl  edb                all  192.168.2.0/24    cert clientcert=1 map=sslusers
```

ステップ 9：以下は、map=sslusers オプションによる pg_hba.conf ファイルに関連する pg_ident.conf ファイル内のユーザ名マップを示しています。これらのユーザ名マップを使用すると、EDB * Plus で接続するときに、データベースユーザ名 edbuser、postgres、または enterprisedb を指定できます。

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

ステップ 10：構成ファイルに変更を加えた後、データベースサーバを再起動します。

4.2 EDB * Plus クライアント用の SSL の構成

データベースサーバで SSL を構成した後、次の手順では、EDB * Plus (JDBC クライアント) の証明書およびキーストアファイルを生成する例を示します。

ステップ 1：データベースサーバのデータサブディレクトリの下にある server.crt および server.key ファイルを使用して、これらのファイルのコピーを作成し、EDB * Plus を実行するホストに移動します。

これらのファイルを目的のディレクトリに保存し、次の手順で生成される信頼できる証明書とキーストアファイルを含めます。推奨される場所は、EDB * Plus を呼び出すホームユーザアカウントの下に .postgresql サブディレクトリを作成することです。したがって、これらのファイルは、EDB * Plus を実行するユーザアカウントの ~/.postgresql ディレクトリの下にあります。

この例では、ファイル edb.crt が server.crt のコピーであり、edb.key が server.key のコピーであると想定しています。

ステップ 2：edb.crt の追加コピーを作成します。

```
$ 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
```

ステップ 3：ファイル edb_root.crt の Distinguished Encoding Rules (DER) フォーマットを作成します。このファイルの生成された DER フォーマットは edb_root.crt.der です。ステップ 4 で使用される keytool プログラムには、ファイルの DER フォーマットが必要です。

```
$ openssl x509 -in edb_root.crt -out edb_root.crt.der -outform der
$ ls -l
```

(次のページに続く)

(前のページからの続き)

```
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
```

ステップ 4: keytool プログラムを使用して、edb_root.crt.der を入力としてキーストアファイル (postgresql.keystore) を作成します。このプロセスにより、Postgres データベースサーバの証明書がキーストアファイルに追加されます。

!!! Note * ファイル名前 postgresql.keystore は、EDB * Plus を呼び出すユーザアカウントのホームディレクトリの下にあるデフォルトのディレクトリ位置 ~/.postgresql postgresql.keystore でアクセスできるようにするために推奨されます。また、ファイル名前のサフィックスは、.keystore ではなく .jks にすることができることにノートしてください（したがって、ファイル名前 postgresql.jks）。ただし、これらの例の残りの部分では、ファイル名前 postgresql.keystore が使用されます。

** Windows のみ: ** パスは %APPDATA%\postgresql\postgresql.keystore です

keytool プログラムは、Java Runtime Environment インストールの bin サブディレクトリの下にあります。

新しいパスワードの入力を求められます。PGSSLCERTPASS 環境変数で指定する必要があるため、このパスワードを保存します。

```
$ /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
```

ステップ 5: ファイル edb.crt および edb.key を入力として使用して、キーストアファイル (postgresql.p12) の PKCS #12 フォーマットを作成します。

!!! Note * ファイル名前 postgresql.p12 は、EDB * Plus を呼び出すユーザアカウントのホームディレクトリの下にあるデフォルトのディレクトリ位置 ~/.postgresql/postgresql.p12 でアクセスできるようにするために推奨されます。

** Windows のみ：**パスは%APPDATA%\postgresql\postgresql.p12 です

新しいパスワードの入力を求められます。PGSSLKEYPASS 環境変数で指定する必要があるため、このパスワードを保存します。

```
$ 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
```

ステップ 6: postgresql.keystore および postgresql.p12 ファイルが ~/.postgresql ディレクトリにない場合、それらをその場所に移動またはコピーします。

** Windows のみ：**ディレクトリは%APPDATA%\postgresql です

ステップ 7: デフォルトの場所 ~/.postgresql を使用しない場合、postgresql.keystore ファイルへのフルパス（ファイル名前を含む）を PGSSLCERT 環境変数で設定し、フルパス（ファイル名前を含む）を EDB * Plus を呼び出す前に、postgresql.p12 ファイルを PGSSLKEY 環境変数で設定する必要があります。

さらに、ステップ 4 で生成されたファイルの名前付けが postgresql.keystore または postgresql.jks でない場合は、PGSSLCERT 環境変数を使用してファイル名前とその場所を指定します。同様に、ステップ 5 で生成されたファイルの名前付けが postgresql.p12 でない場合は、PGSSLKEY 環境変数を使用してファイル名前とその場所を指定します。

4.3 EDB * Plus と Advanced Server データベース間の SSL 接続のリエクスト

SSL 接続を実行するには、オーダーのトピックに対処してください。

EDB * Plus を呼び出すデータベースサーバとクライアントホストの両方に対して、信頼できる証明書とキーストアファイルが生成されています。- データベースサーバの postgresql.conf ファイルには、更新された構成パラメーターが含まれています。- データベースサーバの pg_hba.conf ファイルには、SSL 接続を許可するために必要なエントリーが含まれています。- クライアントホストでは、クライアントの証明書とキーストアファイルがユーザアカウントの ~/.postgresql ディレクトリに配置されているか、環境変数 PGSSLCERT および PGSSLKEY が EDB * Plus を呼び出す前に設定されています。- PGSSLCERTPASS 環境変数にはパスワードが設定されています。- PGSSLKEYPASS 環境変数にパスワードが設定されています

EDB * Plus を呼び出すときは、Using EDB*Plus の connectstring オプションのように、データベースコネクション文字列に?ssl=true パラメータを含めます。

以下は、データベースサーバリモートのホストから EDB * Plus が呼び出される例です。

データベースサーバの `postgresql.conf` ファイルには、次の変更されたパラメーターが含まれています。

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

データベースサーバの `pg_hba.conf` ファイルには、リモートホスト上の EDB * Plus から接続するための次のエントリーが含まれています。

#	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=1

EDB * Plus が呼び出されるリモートホストでは、`user` 名前付けの Linux ユーザアカウントの `~/.postgresql` ディレクトリに証明書とキーストアファイルが含まれています。

```
[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
```

`user` 名前付けのアカウントで Linux にログインすると、EDB * Plus は `ssl=true` パラメータで正常に呼び出されます。

```
$ export PGSSLCERTPASS=keypass
$ export PGSSLKEYPASS=exppass
$ cd /opt/edb/as14/edbplus
$ ./edbplus.sh enterprisedb/password@192.168.2.22:5444/edb?ssl=true
Connected to EnterpriseDB 14.0.0 (192.168.2.22:5444/edb) AS enterprisedb

EDB*Plus: Release 14 (Build 40.0.0)
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.

SQL>
```

または、証明書とキーストアファイルを `~/.postgresql` に配置せずに、別のディレクトリに配置すると、EDB * Plus を次の方法で呼び出すことができます。

```
$ export PGSSLCERT=/home/user/ssl/postgresql.keystore
$ export PGSSLKEY=/home/user/ssl/postgresql.p12
$ export PGSSLCERTPASS=keypass
$ export PGSSLKEYPASS=exppass
$ cd /opt/edb/as14/edbplus
$ ./edbplus.sh enterprisedb/password@192.168.2.22:5444/edb?ssl=true
Connected to EnterpriseDB 14.0.0 (192.168.2.22:5444/edb) AS enterprisedb

EDB*Plus: Release 14 (Build 40.0.0)
Copyright (c) 2008-2021, EnterpriseDB Corporation. All rights reserved.

SQL>
```

どちらの場合も、EDB * Plus へのログに使用されるデータベースユーザ名前は **enterprisedb** であることに注意してください。これは、*Configuring SSL on Advanced Server* のステップ 1 で証明書を作成するときにコモンネームフィールドに指定されるユーザです。

map オプション付きの `pg_hba.conf` ファイルと `pg_ident.conf` ファイルが *Configuring SSL on Advanced Server* のステップ 8 および 9 で説明されているように使用される場合、他のデータベースユーザ名を使用できます。

第5章 Command Summary

</div>

次のセクションには、EDB * Plus コマンドの概要が含まれています。

5.1 受け入れる

ACCEPT コマンドはプロンプトを表示し、ユーザーのキーボード入力を待ちます。ユーザが入力した値は、指定された変数に配置されます。

```
ACC[EPT ] variable
```

次の例では、my_name 名前付けの新しい変数を作成し、John Smith の値を受け入れてから、DEFINE コマンドを使用して値を表示します。

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

5.2 追加

APPEND は、指定されたテキストを SQL バッファの現在の行の末尾に追加する行エディタコマンドです。

```
A[PPEND ] text
```

次の例では、APPEND コマンドを使用して、SELECT コマンドが SQL バッファにビルトインしています。SQL バッファ内で dept と WHERE を 1 つのスペースで区切るオーダーに、APPEND コマンドと WHERE 句の間に 2 つのスペースが配置されることに注意してください。

```
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
```

5.3 変更

CHANGE は、SQL バッファ内の現在の行で検索と置換を実行する行エディタコマンドです。

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

TO/が指定されている場合、現在の行で最初に出現するテキスト **FROM** はテキスト **TO** に変更されます。TO/を省略すると、現在の行で最初に出現したテキスト **FROM** が削除されます。

次のシーケンスのコマンドは、3 行目を現在の行にし、WHERE 句の部門番号を 20 から 30 に変更します。

```
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
```

5.4 クリア

CLEAR コマンドは、SQL バッファの内容を削除するか、COLUMN コマンドで設定されたすべての列定義を削除するか、スクリーンをクリアします。

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

BUFFER | SQL

SQL バッファをクリアします。

COLUMNS

列定義を削除します。

SCREEN

スクリーンをクリアします。オプションが指定されていない場合、これがデフォルト。

5.5 列

COLUMN コマンドは出力フォーマットを制御します。COLUMN コマンドを使用して設定されたフォーマット属性は、現在のセッションの間のみ有効です。

COL[UMN]

```
* [ column
*   { CLE[AR ] |
*     { FOR[MAT ] spec |
*       HEA[DING ] text |
*       { OFF | ON }
*     } [...]
*   }
* ]
```

COLUMN コマンドが後続のオプションなしで指定されている場合、セッションで有効な現在の列のフォーマットオプションが表示されます。

COLUMN コマンドの後に列名前の後に次のいずれかが続く場合があります。

1. 他のオプションはありません 2. 。FORMAT、HEADING、および OFF または ON のいずれかの組み合わせ

column

後続の列書式設定オプションが適用される表の列の名前。column の後に他のオプションがない場合、column の現在の列フォーマットオプションが表示されます（存在する場合）。

CLEAR

CLEAR オプションは、すべてのフォーマットオプションを column のデフォルトに戻します。CLEAR オプションを指定する場合、指定する唯一のオプションでなければなりません。

spec

column に適用されるフォーマット仕様。文字列の場合、spec のフォーマットは次のとおりです。

n

n は、データを表示する列幅を文字単位で指定する正の整数です。n を超えるデータは、指定された列幅でラップします。

数値列の場合、spec は次の要素で構成されます。

Element	Description
\$	先頭のドル記号を表示します。
,	指定された位置にコンマを表示します。
.	10 進数の位置を示します。
0	先頭のゼロを表示します。
9	表示する有効数字数。

次のページに続く

表 1 – 前のページからの続き

Element	Description
	NA

フォーマットのオーバーフローにより有効数字の損失が発生した場合、すべての # が表示されます。

text

column の列見出しに使用されるテキスト。

OFF | ON

OFF が指定されている場合、フォーマットオプションはデフォルトに戻りますが、セッション内では引き続き使用できます。ON が指定されている場合、セッション内の column に対して以前の COLUMN コマンドで指定されたフォーマットオプションが再アクティブ化されます。

次の例は、job 列の表示幅を変更した場合の効果を示しています。

```
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;
```

```
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
```

(次のページに続く)

(前のページからの続き)

```

7900 JAMES      CLERK
7902 FORD      ANALY
*              ST
* 7934 MILLER   CLERK

```

14 rows retrieved.

次の例では、sal 列にフォーマットを適用します。

```

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;

```

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	

(次のページに続く)

(前のページからの続き)

```
* 7934 MILLER      CLERK      $1,300.00
```

```
14 rows retrieved.
```

5.6 接続

データベースコネクションを別のユーザに変更するか、別のデータベースに接続します。CONNECT コマンドに続くパラメーターの間に空白文字があってはなりません。

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

どこで：

username は、データベースに接続するためのデータベースユーザー名です。

password は、指定された **username** に関連付けられたパスワード。**password** が提供されていないが、認証にパスワードが必要な場合、最初に EDB * Plus を呼び出す Linux オペレーティングシステムアカウントのホームディレクトリ（または Windows の場合は%APPDATA%\postgresql\ディレクトリ）でパスワードファイルが検索されます。次に、PGPASSFILE 環境変数で指定された場所。パスワードファイルは、Linux ホストでは .pgpass、Windows ホストでは pgpass.conf です。以下は、Windows ホストでの例です。

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

パスワードファイルが見つからない場合、または EDB * Plus 接続パラメーターにマッチングするエントリーがない場合、EDB * Plus はパスワードのプロンプトを求めます。パスワードファイルの詳細については、次の PostgreSQL コア文書を参照してください。

< <https://www.postgresql.org/docs/current/libpq-pgpass.html> >

注：パスワードが不要な場合、trust 認証メソッドが pg_hba.conf ファイルで指定されている場合など、EDB * Plus はパスワードのプロンプトを求めません。pg_hba.conf ファイルと認証方法の詳細については、< <https://www.postgresql.org/docs/current/auth-pg-hba-conf.html> > の PostgreSQL コア文書を参照してください。

connectstring はデータベースコネクション文字列です。データベースコネクション文字列の詳細については、Using EDB*Plus を参照してください。

variable は、データベースコネクション文字列を含む login.sql ファイルで定義された変数。login.sql ファイルは、Advanced Server ホームディレクトリサブディレクトリにあります。

次の例では、データベースコネクションが、ユーザー名 smith のポート 5445 のローカルホスト上のデータベース edb に変更されます。

```
SQL> CONNECT smith/mypassword@localhost:5445/edb
Disconnected from EnterpriseDB Database.
Connected to EnterpriseDB 14.0.0 (localhost:5445/edb) AS smith
```

上記のセッション内から、接続はユーザー名 **enterprisedb** に変更されます。また、ホストのデフォルトはローカルホスト、ポートのデフォルトはノート（以前に使用したポートとは異なります）、およびデータベースのデフォルトは **edb** であることに注意してください。

```
SQL> CONNECT enterprisedb/password
Disconnected from EnterpriseDB Database.
Connected to EnterpriseDB 14.0.0 (localhost:5444/edb) AS enterprisedb
```

5.7 定義

DEFINE コマンドは、**ユーザ変数**（**置換変数**とも呼ばれる）の値を作成または置換します。

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

DEFINE コマンドがパラメーターなしで指定された場合、現在の変数とその値がすべて表示されます。

DEFINE **variable** が指定されている場合、**variable** のみがその値とともに表示されます。

DEFINE **variable** = **text** は **text** を **variable.text** に割り当て、オプションで単一引用符または二重引用符で囲むことができます。**text** にスペース文字が含まれる場合は、引用符を使用する必要があります。

次の例では、**dept** と **name** の 2 つの変数を定義しています。

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

!!! Note * 変数 EDB は、Advanced Server ホームディレクトリサブディレクトリにある **login.sql** ファイルから読み取られます。

5.8 DEL

DEL は、SQL バッファから 1 つ以上の行を削除する行エディタコマンドです。

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

パラメーターは、SQL バッファから削除する行を指定します。2 つのパラメーターは、削除する行のレンジのスタートと終了を指定します。DEL コマンドがパラメーターなしで指定された場合、現在の行は削除されます。

n

n は **n** 行目を表す整数です

n m

n および m は、 m が n th から m th 行までを表す n より大きい整数です。

*

現在の行

LAST

最終行

次の例では、列 `sal` および `comm` をそれぞれ含む 5 行目と 6 行目が、SQL バッファの `SELECT` コマンドから削除されます。

```
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
```

5.9 説明

`DESCRIBE` コマンドは次を表示します。

- テーブルまたはビューの列、列のデータ型、および列の長さのリスト
- プロシージャまたは関数のパラメータのリスト
- パッケージのプロシージャと関数、およびそれぞれのパラメータのリスト

`DESCRIBE` コマンドは、シノニムによって参照されるデータベースオブジェクトの構造も表示します。構文は次のとおりです。

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

`schema`

記述されるオブジェクトを含むスキーマの名前。

object

表示するテーブル、ビュー、プロシージャ、ファンクション、またはパッケージの名前、またはオブジェクトのシノニム。

5.10 切断

DISCONNECT コマンドは現在のデータベースコネクションを閉じますが、EDB * Plus は終了しません。

```
DISC[ONNECT ]
```

5.11 編集

EDIT コマンドは、外部エディタを呼び出して、オペレーティングシステムファイルまたは SQL バッファの内容を編集します。

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

filename[.ext]

filename は、外部エディタで開くファイルの名前です。ext はファイル名拡張子です。ファイル名の拡張子が sql の場合、filename を指定するときに .sql 拡張子を省略できます。EDIT は、拡張子なしで指定されたファイル名に対して常に .sql 拡張子を想定しています。EDIT コマンドで filename パラメータが省略されている場合、SQL バッファの内容がエディタに取り込まれます。

5.12 実行

EXECUTE コマンドは、EDB * Plus から SPL プロシージャを実行します。

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

spl_procedure

実行する SPL プロシージャの名前。

parameters

パラメーターのコンマ区切りリスト。パラメータがない場合は、空のカッコのペアをオプションで指定できます。

5.13 終了

EXIT コマンドは、EDB *Plus セッションを終了し、オペレーティングシステムに制御を返します。QUIT は EXIT の同義語です。パラメーターを指定しないことは、EXIT SUCCESS COMMIT と同等です。

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

SUCCESS、FAILURE、および WARNING について、それぞれ成功したオペレーション、失敗、またはワーニングを示すオペレーティングシステム依存のリターンコードを返します。デフォルトは SUCCESS です。

value

リターンコードとして返される整数値。

variable

値が戻りリターンコードとして返される DEFINE コマンドで作成された変数。

COMMIT | ROLLBACK

COMMIT が指定されている場合、コミットされていない更新は終了時にコミットされます。ROLLBACK が指定されている場合、コミットされていない更新は終了時にロールバックされます。デフォルトは COMMIT です。

5.14 GET

GET コマンドは、指定されたファイルの内容を SQL バッファにロードします。

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

filename[.ext]

filename は、SQL バッファにロードするファイルの名前です。**ext** はファイル名拡張子です。ファイル名の拡張子が **sql** の場合、**filename** を指定するときに **.sql** 拡張子を省略できます。GET は、拡張子なしで指定されたファイル名に対して常に **.sql** 拡張子を想定しています。

LIST | NOLIST

LIST が指定されている場合、ファイルのロード後に SQL バッファのコンテンツが表示されます。NOLIST が指定されている場合、リストは表示されません。デフォルトは LIST です。

5.15 ヘルプ

HELP コマンドは、トピックのインデックスまたは特定のトピックのヘルプを取得します。マーク (?) は、HELP を指定することと同義です。

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

INDEX

利用可能なトピックのインデックスを表示します。

topic

ヘルプが必要な特定のトピックの名前（EDB * Plus コマンドなど）。

5.16 ホスト

HOST コマンドは、EDB * Plus からオペレーティングシステムコマンドを実行します。

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

os_command

実行されるオペレーティングシステムコマンド。オペレーティングシステムコマンドを指定しない場合、EDB * Plus は実行を一時停止し、新しいシェルプロンプトを開きます。シェルが終了すると、EDB * Plus は実行を再開します。

5.17 入力

INPUT 行エディタコマンドは、現在の行の後にテキストの行を SQL バッファに追加します。

```
I[INPUT ] text
```

次のシーケンスの INPUT コマンドは、SELECT コマンドを構成します。

```
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
```

5.18 リスト

LIST は、SQL バッファの内容を表示する行エディタコマンドです。

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

バッファには、EDB *Plus コマンドの履歴は含まれません。

n

n は、バッファの行番号を表します。

n m

n m は、**n** と **m** の間の行のリストを表示します。

n *

n * は、**n** 行から現在の行までのレンジの行のリストを表示します。

n L[AST]

n L[AST] は、レンジ行からバッファの最後の行までの行のリストを表示します。

***** は現在の行を表示します。

*** n**

*** n** は、現在の行から **n** 行までのレンジの行のリストを表示します。

*** L[AST]**

*** L[AST]** は、現在の行から最後の行までのレンジの行のリストを表示します。

L[AST]

L[AST] は最後の行を表示します。

5.19 パスワード

PASSWORD コマンドを使用して、データベースのパスワードを変更します。

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

PASSWORD コマンドを使用して別のユーザーのパスワードを変更するには、十分な特権が必要です。次の例は、PASSWORD コマンドを使用して、acctg 名前付けのユーザのパスワードを変更する方法を示しています。

```
SQL> PASSWORD acctg
Changing password for acctg
*   New password:
```

(次のページに続く)

(前のページからの続き)

```
*      New password again:
Password successfully changed.
```

5.20 一時停止

PAUSE コマンドはメッセージを表示し、ユーザが ENTER を押すのを待ちます。

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

`optional_text` は、ユーザに表示されるテキストを指定します。`optional_text` を省略すると、Advanced Server は 2 つの空白行を表示します。`optional_text` 文字列を二重引用符で囲むと、引用符が出力に含まれます。

5.21 プロンプト

PROMPT コマンドは、続行する前にユーザにメッセージを表示します。

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

`message_text` は、ユーザに表示されるテキストを指定します。出力に引用符を含めるには、文字列を二重クォートで囲みます。

5.22 終了

QUIT コマンドは、セッションを終了し、オペレーティングシステムに制御を返します。QUIT は EXIT の同義語です。

```
QUIT

[SUCCESS | FAILURE | WARNING | value | sub_variable]

[COMMIT | ROLLBACK]
```

デフォルト値は QUIT SUCCESS COMMIT です。

5.23 備考

REMARK を使用して、スクリプトにコメントを含めます。

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

次の規則を使用してコメントを含めることもできます。

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

5.24 保存

SAVE コマンドを使用して、SQL バッファをオペレーティングシステムファイルに書き込みます。

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

file_name

file_name は、バッファの内容が書き込まれるファイルの名前（パスを含む）を指定します。ファイル拡張子を指定しない場合、ファイル名前の末尾に **.sql** が追加されます。

CREATE

新しいファイルを作成するには、**CREATE** キーワードを含めます。指定した名前のファイルがまだ存在しない場合にのみ、新しいファイルが作成されます。これがデフォルト。

REPLACE

REPLACE キーワードを含めて、Advanced Server が既存のファイルを上書きするように指定します。

APPEND

APPEND キーワードを含めて、Advanced Server が指定されたファイルの最後に SQL バッファの内容を追加することを指定しファイル。

次の例では、SQL バッファの内容を **temp** ディレクトリにある **example.sql** 名前付けのファイルに保存します。

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

5.25 セット

SET コマンドを使用して、EDB *Plus の動作を制御するセッションレベル変数の値を指定します。次の形式の SET コマンドが有効です。

オートコミットを設定

SET AUTOCOMMIT コマンドを使用して、Advanced Server トランザクションのコミット動作を指定します。

```
SET AUTO[COMMIT]
```

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

EDB *Plus は常に DDL ステートメントを自動的にコミットすることにノートしてください。

ON

AUTOCOMMIT の動作をオンにするには、ON を指定します。

OFF

AUTOCOMMIT の動作をオフにするには、OFF を指定します。

IMMEDIATE

IMMEDIATE は ON と同じ効果があります。

statement_count

statement_count の値を含めて、指定した数の成功した SQL ステートメントの後にコミットを発行するように EDB *Plus に指示します。

カラムセパレータを設定

SET COLUMN SEPARATOR コマンドを使用して、Advanced Server が列の間に表示するテキストを指定します。

```
SET COLSEP column_separator
```

column_separator のデフォルト値は単一スペースです。

エコーの設定

SET ECHO コマンドを使用して、SQL および EDB *Plus スクリプトステートメントを実行時に画面に表示するかどうかを指定します。

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

デフォルト値は OFF です。

フィードバックを設定

SET FEEDBACK コマンドは、SQL ステートメントの実行後のインタラクティブ情報の表示を制御します。

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

row_threshold

row_threshold に整数値を指定します。row_threshold を 0 に設定することは、FEEDBACK を OFF に設定することと同じです。row_threshold を 1 と等しく設定すると、FEEDBACK が ON に効果的に設定されます。

フラッシングセット

SET FLUSH コマンドを使用して、ディスプレイのバッファリングを制御します。

SET FLU[SH] {ON | OFF}

表示バッファリングを有効にするには、FLUSH を OFF に設定します。バッファリングを有効にすると、スクリーンにバインドされたメッセージは、スクリプトが完了するまで表示されない場合があります。FLUSH を OFF に設定すると、パフォーマンスが向上することにノートしてください。

表示バッファリングを無効にするには、FLUSH を ON に設定します。バッファリングを無効にすると、スクリーンにバインドされたメッセージがすぐに表示されます。

見出しの設定

SET HEADING 変数を使用して、Advanced Server が SELECT ステートメントの列見出しを表示するかどうかを指定します。

SET HEA[DING] {ON | OFF}

ヘッドセパレーターのセット

SET HEADSEP コマンドは、COLUMN HEADING コマンドで使用する新しい見出し区切り文字を設定します。デフォルトは'|'です。

SET HEADS[EP]

ラインサイズの設定

SET LINESIZE コマンドを使用して、行の幅を文字で指定します。

SET LIN[ESIZE] width_of_line

width_of_line

width_of_line のデフォルト値は 132 です。

**** SET NEWPAGE ****

SET NEWPAGE コマンドを使用して、改ページ後に印刷される空白行の数を指定します。

SET NEWP[AGE] lines_per_page

lines_per_page

lines_per_page のデフォルト値は 1 です。

**** NULL を設定****

NULL コマンドを使用して、NULL 列の値が出力バッファに表示されるときにユーザに表示される文字列を指定します。

```
SET NULL null_string
```

ページサイズの設定

SET PAGESIZE コマンドを使用して、ページに収まる印刷行数を指定します。

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

line_count パラメータを使用して、ページごとの行数を指定します。

** SET SQLCASE **

SET SQLCASE コマンドは、サーバーに送信される SQL ステートメントを大文字に変換するか小文字に変換するかを指定します。

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

UPPER

UPPER を指定して、コマンドテキストを大文字に変換します。

LOWER

コマンドテキストを小文字に変換するには、LOWER を指定します。

MIXED

MIXED を指定して、SQL コマンドの大文字と小文字を変更しません。デフォルトは MIXED です。

一時停止を設定

SET PAUSE コマンドは、スクリプトに含まれている場合に最も役立ちます。コマンドはプロンプトを表示し、ユーザが Return を押すのを待ちます。

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

SET PAUSE が ON の場合、各コマンドが実行される前にメッセージ Hit ENTER to continue…が表示されます。

スペースの設定

SET SPACE コマンドを使用して、列間に表示するスペースの数を指定します。

```
SET SPACE number_of_spaces
```

** SET SQLPROMPT **

SET SQLPROMPT を使用して、ユーザーインタラクティブプロンプトの値を設定します。

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

デフォルトでは、SQLPROMPT は"SQL> "に設定されます

端末の設定

SET TERMOUT コマンドを使用して、コマンド出力を画面に表示するかどうかを指定します。

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

タイミングの設定

SET TIMING コマンドは、Advanced Server が実行された後、各 SQL ステートメントの実行時間を表示するかどうかを指定します。

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

**** TRIMSPPOOL を設定****

SET TRIMSPPOOL コマンドを使用して、SPOOL コマンドで指定された出力ファイルの各行から末尾のスペースを削除します。

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

デフォルト値は OFF です。

確認を設定

置換変数が検出されたときに、SQL ステートメントの古い値と新しい値の両方を表示するかどうかを指定します。

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

5.26 ショー

SHOW コマンドを使用して、現在のパラメータ値を表示します。

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

ALL キーワードを含めて、現在のパラメータ設定を表示します。

```
SQL> SHOW ALL
autocommit      OFF
colsep          " "
define          "&"
echo            OFF
FEEDBACK ON for 6 row(s).
flush           ON
heading         ON
headsep        "|"
```

(次のページに続く)

(前のページからの続き)

```

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"

```

または、SHOW コマンドに `parameter_name` を含めることにより、特定のパラメータ設定を表示します。

```

SQL> SHOW VERSION
VERSION is "14.0.0"

```

5.27 スプール

SPPOOL コマンドは、ディスプレイからの出力をファイルに送信します。

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

`output_file` パラメータを使用して、出力ファイルのパス名前を指定します。

5.28 開始

START コマンドを使用して、EDB * Plus スクリプトファイルを実行します。START は、@コマンドの別名です。

```
STA[RT] script_file
```

`script_file` パラメータでスクリプトファイルの名前を指定します。

5.29 未定義

UNDEFINE コマンドは、DEFINE コマンドによって作成されたユーザ変数を消去します。

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

variable_name パラメータを使用して、変数の名前を指定します。

5.30 いつでも SQLERROR

WHENEVER SQLERROR コマンドは、SQL エラーまたは PL/SQL ブロックエラーのエラーハンドリングを提供します。構文は次のとおりです。

```
WHENEVER SQLERROR
* {CONTINUE[COMMIT|ROLLBACK|NONE]
* |EXIT[SUCCESS|FAILURE|WARNING|n|sub_variable]
* [COMMIT|ROLLBACK]}
```

SQL コマンドまたは PL/SQL ブロックの実行中に Advanced Server でエラーが発生した場合、EDB * Plus は WHENEVER SQLERROR コマンドで指定されたアクションを実行します。

CONTINUE 句を含めて、続行する前に指定されたアクションを実行するように EDB * Plus に指示します。

COMMIT 句を含めて、終了または続行する前に EDB * Plus に現在のトランザクションを COMMIT に指示します。

ROLLBACK 句を含めて、終了または続行する前に EDB * Plus に現在のトランザクションを ROLLBACK に指示します。

NONE 句を含めて、トランザクションをコミットまたはロールバックせずに続行するように EDB * Plus に指示します。

EXIT 句を含めて、指定されたアクションを実行し、エラーが発生した場合は終了するように EDB * Plus に指示します。

次のオプションを使用して、終了前に EDB * Plus が結果ステータスコードを指定します。

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

EDB * Plus は置換変数をサポートしていますが、バインド変数はサポートしていないことにノートしてください。