



OCCL Connector

リリース *12.1.2.1*

EDB Postgres™ Advanced Server OCL Connector Guide

2020 年 06 月 30 日

目次

第 1 章	新機能	3
第 2 章	サポートされているプラットフォーム	5
第 3 章	OpenClientLibrary	7
3.1	OCL コネクタのインストールと設定	8
3.1.1	RPM パッケージを使用したコネクタのインストール	8
3.1.2	SLES12 ホストへのコネクタのインストール	11
3.1.3	Debian または Ubuntu ホストへの DEB パッケージのインストール	13
3.1.4	グラフィカルインストーラーを使用したコネクタのインストール	14
3.2	接続文字列の形成	16
3.3	プログラムのコンパイルとリンク	20
3.4	RefCursorSupport	21
3.5	OCL 関数リファレンス	24
3.5.1	接続、承認、初期化機能	24
3.5.2	ハンドルおよび記述子関数	26
3.5.3	関数のバインド、定義、説明	30
3.5.4	ステートメント関数	30
3.5.5	トランザクション関数	30
3.5.6	XA 関数	31
3.5.7	日付および日時関数	32
3.5.8	インターバル関数	33
3.5.9	数値関数	34
3.5.10	文字列関数	36
3.5.11	カートリッジサービスとファイル I/O インターフェイス関数	36
3.5.12	LOB 関数	36
3.5.13	その他の機能	37
3.5.14	サポートされているデータ型	38
3.6	OCL エラーコード-リファレンス	39
3.7	マルチスレッドのサポート	41
第 4 章	OCL トレースの生成	43

第 5 章	セキュリティと暗号化	45
5.1	SSL の使用	45
5.2	スクラムの互換性	46
第 6 章	結論	47
索引		49

OCL コネクタは、OracleCallInterface と同様の API を提供します。OracleCallInterface を使用するように作成されたアプリケーションは、EnterpriseDB の OCL コネクタを使用して再コンパイルし、AdvancedServer データベースサーバーとやり取りすることができます。

このガイドでは、次のインストールと使用方法について説明します。

- コネクタの取り付け方法。
- Oracle スタイルの接続文字列を作成する方法。
- プログラムをコンパイルしてリンクする方法。

このガイドには、AdvancedServer がサポートする機能のリファレンスセクションも含まれています。

注釈: EnterpriseDB は、OracleRealApplicationClusters (RAC) および OracleExadata での OpenClientLibrary の使用をサポートしていません。前述の Oracle 製品は、この EnterpriseDB 製品では評価も認定もされていません。

第 1 章

新機能

AdvancedServerOCL コネクタ 12.1.2.1 を作成するために、次の機能が追加されました。

- EDBOCL コネクタはマルチスレッドサポートを提供するようになりました。
- EDBOCL コネクタが EDBPostgresAdvancedServer12 をサポートするようになりました。
- EDBOCL コネクタは、WindowsServer2019 プラットフォームでもサポートされるようになりました。

第 2 章

サポートされているプラットフォーム

AdvancedServerOCL コネクタは、AdvancedServer バージョン 9.4 以降で認定されています。OCL コネクタネイティブパッケージは、次の 64 ビットプラットフォームでサポートされています。

- RedHatEnterpriseLinux (x86_64) 6.x および 7.x
- CentOS (x86_64) 6.x および 7.x
- OELLinux6.x および 7.x
- RHEL または CentOS7.x を実行する PPC-LE8
- SLES12.x
- Debian9.x
- Ubuntu18.04

OCL コネクタのグラフィカルインストーラーは、次の 64 ビット Windows プラットフォームでサポートされています。

- Windows サーバー 2019
- Windows サーバー 2016
- Windows サーバー 2012R2
- ウィンドウズ 10
- ウィンドウズ 8
- Windows7

OCL コネクタのグラフィカルインストーラーは、次の 32 ビット Windows プラットフォームでサポートされています。

- ウィンドウズ 10

- ウィンドウズ 8
- Windows7

第 3 章

OpenClientLibrary

OpenClientLibrary は、OracleCallInterface とのアプリケーションの相互運用性を提供します。以前はロックインされていたアプリケーションが、EDBPostgresAdvancedServer または Oracle データベースのいずれかで動作し、アプリケーションコードをほとんどまたはまったく変更する必要がありません。

次の図は、OpenClientLibrary と OracleCallInterface アプリケーションスタックを比較しています。

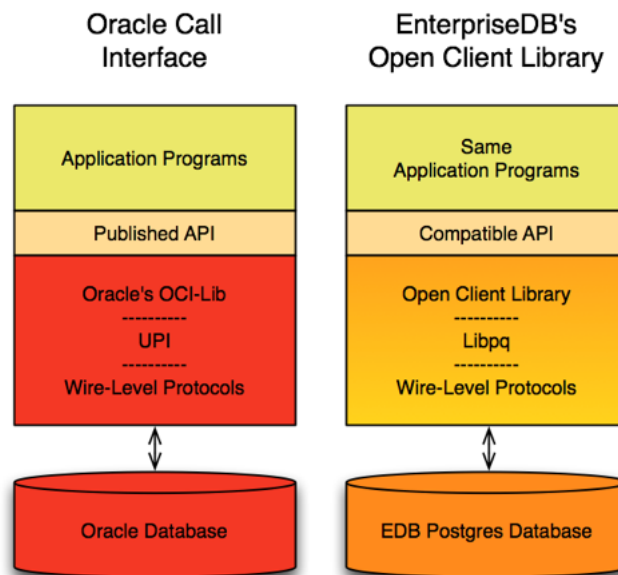


図 1 *OracleCallInterface* との比較

OpenClientLibrary の EnterpriseDB 実装は C で書かれています。

3.1 OCL コネクタのインストールと設定

RPM パッケージ、ネイティブパッケージ、またはグラフィカルインストーラーを使用して、OCL コネクタをインストールまたは更新できます。

3.1.1 RPM パッケージを使用したコネクタのインストール

OCL Connector 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します:

- RHEL または CentOS7 の場合:

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

- RHEL または CentOS8 の場合:

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

注釈: (/etc/yum.repos.d にある) CentOS-Base.repo ファイルの [extras] リポジトリ定義を有効にする必要があるかもしれません。

EnterpriseDB 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、次の URL にアクセスしてください。

<https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%202004-09-2019.pdf>

リポジトリの資格情報を受け取ったら、次のことができます。

1. リポジトリ構成ファイルを作成します。
2. ユーザー名とパスワードを指定して、ファイルを変更します。
3. edb-oci をインストールします。

リポジトリ設定ファイルの作成

リポジトリ構成ファイルを作成するには、スーパーユーザー特権を想定して、次のコマンドを呼び出します。

- RHEL または CentOS7 の場合:

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

- RHEL または CentOS8 の場合:

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

リポジトリ設定ファイルの名前は `edb.repo` です。ファイルは `/etc/yum.repos.d` にあります。

ファイルを変更し、ユーザー名とパスワードを提供します

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

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

OCL コネクタのインストール

After saving your changes to the configuration file, you can use the `yum install` command to install OCL Connector. For example, the following command installs OCL Connector:

- RHEL または CentOS7 の場合:

```
yum install edb-oci
yum install edb-oci-devel
```

- RHEL または CentOS8 の場合:

```
dnf install edb-oci
dnf install edb-oci-devel
```

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` はローカルサーバーにキーをインポートする許可を要求する場合があります。プロンプトが表示され、パッケージ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` を入力し、Return を押して続行します。

インストール中に、`yum` は解決できない依存関係に遭遇する可能性があります。含まれている場合は、手動で解決する必要がある必要な依存関係のリストが表示されます。

RPM インストールの更新

既存の OCL Connector RPM インストールがある場合は、`yum` または `dnf` を使用してリポジトリ構成ファイルをアップグレードし、より新しい製品バージョンに更新できます。 `edb.repo` ファイルを更新するには、スーパーユーザー権限を引き受け、次のように入力します:

- RHEL または CentOS7 の場合:

```
yum upgrade edb-repo
```

- RHEL または CentOS8 の場合:

```
dnf upgrade edb-repo
```

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

- RHEL または CentOS7 の場合:

```
yum upgrade edb-oci
```

```
yum upgrade edb-oci-devel
```

- RHEL または CentOS8 の場合:

```
dnf upgrade edb-oci
```

```
dnf upgrade edb-oci-devel
```

3.1.2 SLES12 ホストへのコネクタのインストール

zypper パッケージマネージャーを使用して、SLES12 ホストにコネクタをインストールできます。zypper は、パッケージをインストールするときにパッケージの依存関係を満たすよう試みますが、EnterpriseDB でホストされていない特定のリポジトリへのアクセスが必要です。コネクタをインストールする前に、次のコマンドを使用して EnterpriseDB リポジトリ構成ファイルを SLES ホストに追加します。

```
zypper addrepo https://zypp.enterprisedb.com/suse/epas12-sles.repo
zypper addrepo https://zypp.enterprisedb.com/suse/epas-sles-tools.repo
zypper addrepo
https://zypp.enterprisedb.com/suse/epas-sles-dependencies.repo
```

各コマンドはリポジトリ設定ファイルを /etc/zypp/repos.d ディレクトリに作成します。ファイルの名前は次のとおりです。

- edbas12suse.repo
- edbasdependencies.repo
- edbastools.repo

リポジトリ設定ファイルを作成したら、zypper refresh コマンドを使用して SLES ホストのメタデータを更新し、EnterpriseDB リポジトリを含めます。

```
/etc/zypp/repos.d # zypper refresh
Repository 'SLES12-12-0' is up to date.
Repository 'SLES12-Pool' is up to date.
Repository 'SLES12-Updates' is up to date.
Retrieving repository 'EDB Postgres Advanced Server 12 12 - x86_64'
metadata -----[\]

Authentication required for
'https://zypp.enterprisedb.com/12/suse/suse-12-x86_64'
User Name:
Password:

Retrieving repository 'EDB Postgres Advanced Server 12 12 - x86_64'
metadata.....[done]
Building repository 'EDB Postgres Advanced Server 12 12 - x86_64'
cache.....[done]
All repositories have been refreshed.
...
```

User Name と Password の入力を求められたら、EnterpriseDB リポジトリの接続認証情報を入力します。資格情報が必要な場合は、**EnterpriseDB** にお問い合わせください<[<https://www.enterprisedb.com/repository-access-request/>](https://www.enterprisedb.com/repository-access-request)>`_。

EDBPostgresAdvancedServer またはサポートコンポーネントをインストールする前に、SUSEConnect と SUSEPackageHub 拡張を SLES ホストに追加し、ホストを SUSE に登録して、SUSE リポジトリへのアクセスを許可する必要があります。次のコマンドを使用します。

```
zypper install SUSEConnect
SUSEConnect -p PackageHub/12/x86_64
SUSEConnect -p sle-sdk/12/x86_64
```

SUSE ホストの登録の詳細については、`[SUSEWeb](https://www.suse.com/support/kb/doc/?id=7016626) サイトにアクセスしてください。
<<https://www.suse.com/support/kb/doc/?id=7016626>>`。

次に、zypper ユーティリティを使用してコネクタをインストールします。

```
zypper install edb-oci
zypper install edb-oci-devel
```

3.1.3 Debian または Ubuntu ホストへの DEB パッケージのインストール

Debian または Ubuntu ホストに DEB パッケージをインストールするには、EnterpriseDB 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。リポジトリの認証情報をリクエストするには、`EDBWeb サイトにアクセスしてください<<https://www.enterprisedb.com/repository-access-request>>`。

次の手順では、EnterpriseDBapt リポジトリを使用して DEB パッケージをインストールする方法について説明します。コマンドを使用する場合、username と password を EnterpriseDB によって提供される資格情報に置き換えます。

1. スーパーユーザー権限を想定します。

```
sudo su -
```

2. EnterpriseDB リポジトリを設定します。

```
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'
```

3. 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 install apt-transport-https
```

4. EBD 署名鍵を追加します。

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

5. リポジトリのメタデータを更新します。

```
apt-get update
```

6. DEB パッケージをインストールします。

```
apt-get install edb-oci
```

```
apt-get install edb-oci-dev
```

3.1.4 グラフィカルインストーラーを使用したコネクタのインストール

EnterpriseDB コネクタインストールウィザードを使用して、システムに OCL コネクタを追加できます。ウィザードは `EnterpriseDBAdvancedDownloads` ページから入手できます<<https://www.enterprisedb.com/advanced-downloads>>`_。

このセクションでは、インストールウィザードを使用して Windows システムにコネクタをインストールする方法を示します。(インストーラーをダウンロードし、インストーラーアイコンを右クリックして、コンテキストメニューから Run As Administrator を選択します)

Language Selection ポップアップが開いたら、インストール言語を選択し、OK をクリックして Setup ウィンドウに進みます。

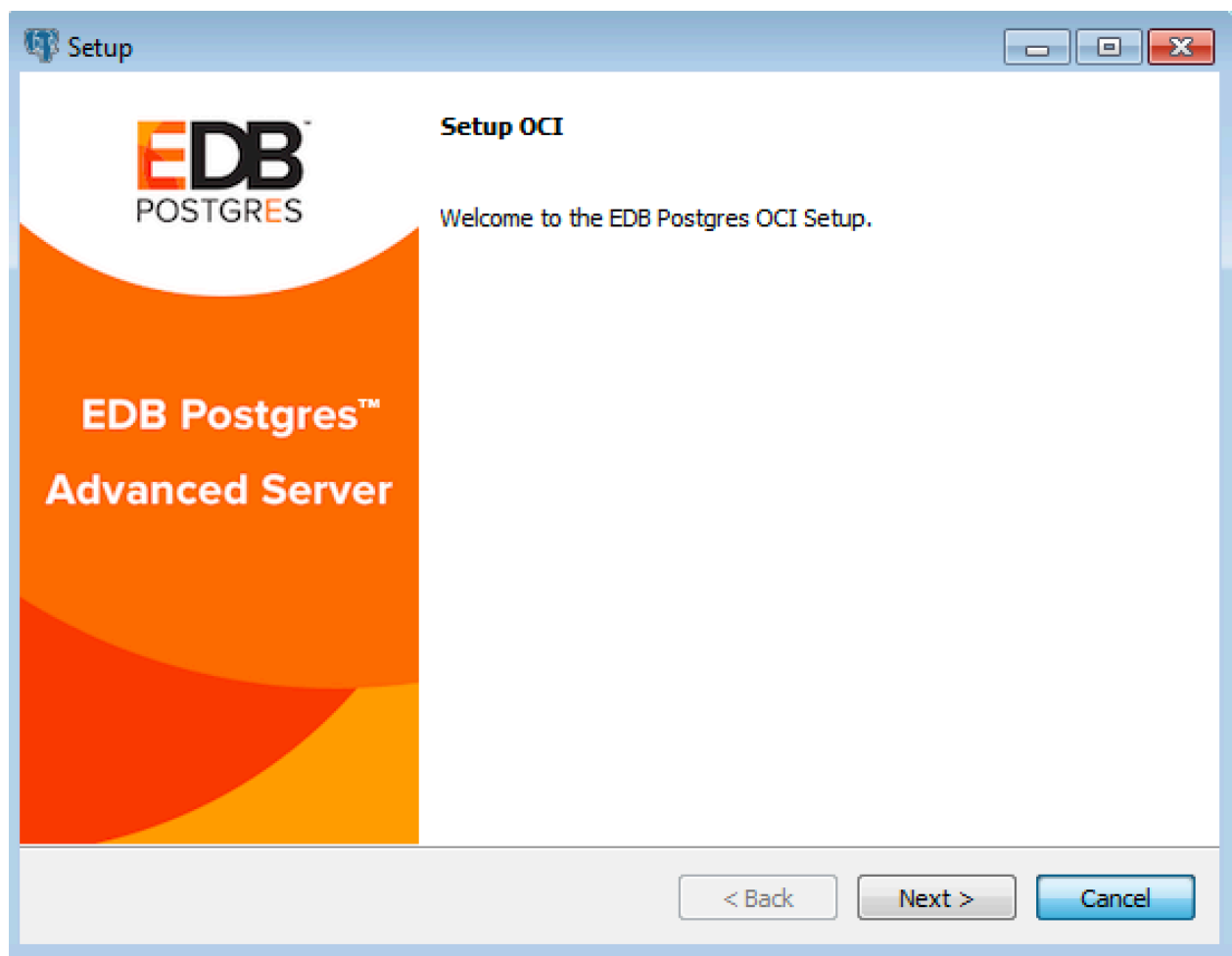


図 2 OCL コネクタインストールウィザード

Next をクリックして続行します。

Installation Directory ダイアログを使用してコネクタをインストールするディレクトリを指定し、Next をクリックして続行します。

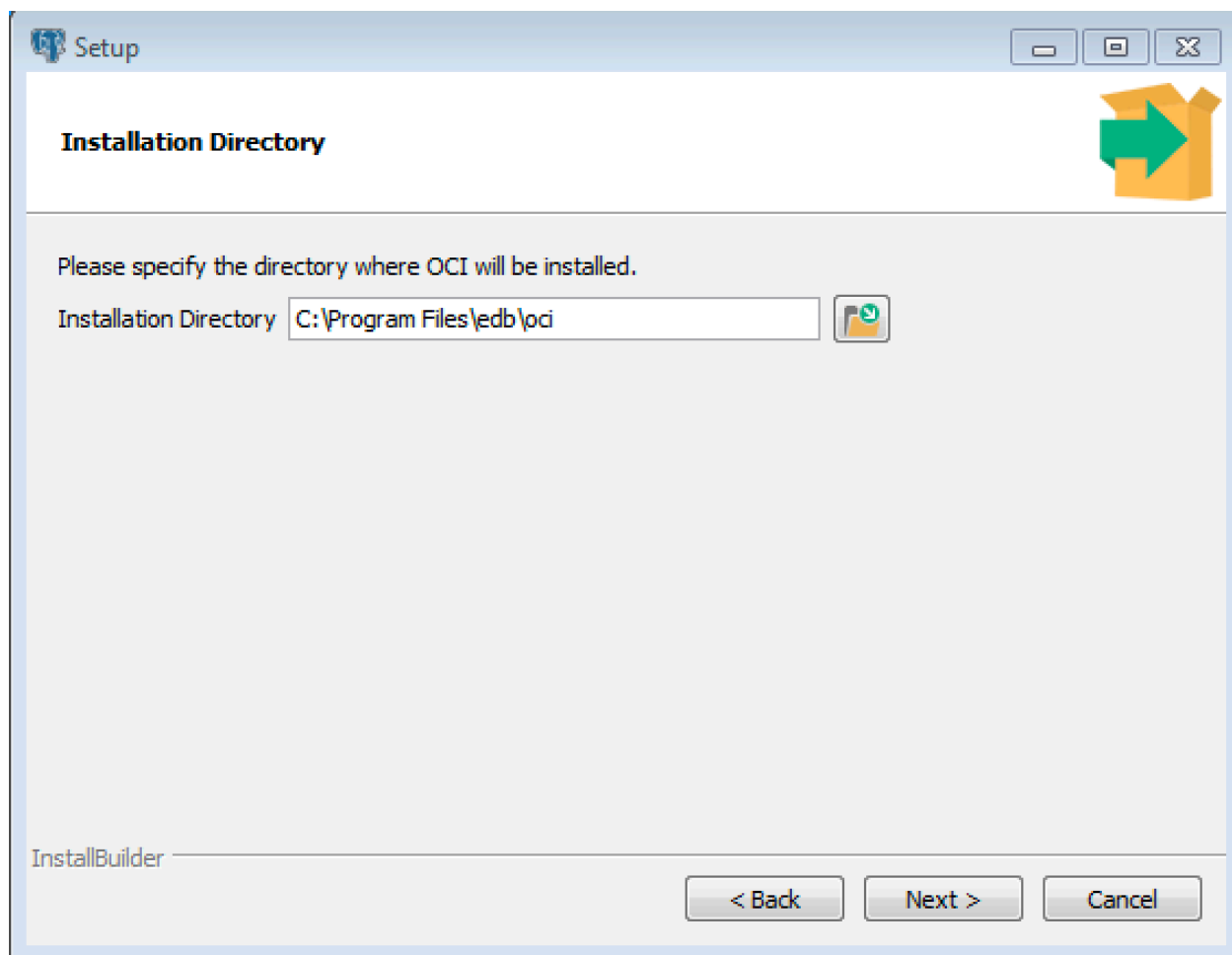


図3 インストールダイアログ

Ready to Install ダイアログの Next をクリックしてインストールを開始します。ポップアップダイアログは、インストールウィザードの進行状況を確認します。

セットアップが完了したことがウィザードから通知されたら、Finish ボタンをクリックしてダイアログを終了します。

StackBuilderPlus を使用して、既存の AdvancedServer インストールでコネクタを追加または更新することもできます。StackBuilderPlus を開くには、Windows の Apps メニューから、または Linux の Applications メニューから StackBuilderPlus を選択します。

StackBuilderPlus が開いたら、画面の指示に従います。ツリーコントロールの Database Drivers ノードから EnterpriseDB OCI Connector オプションを選択します。

画面上のウィザードの指示に従って、EnterpriseDB コネクタのインストールを追加または更新します。

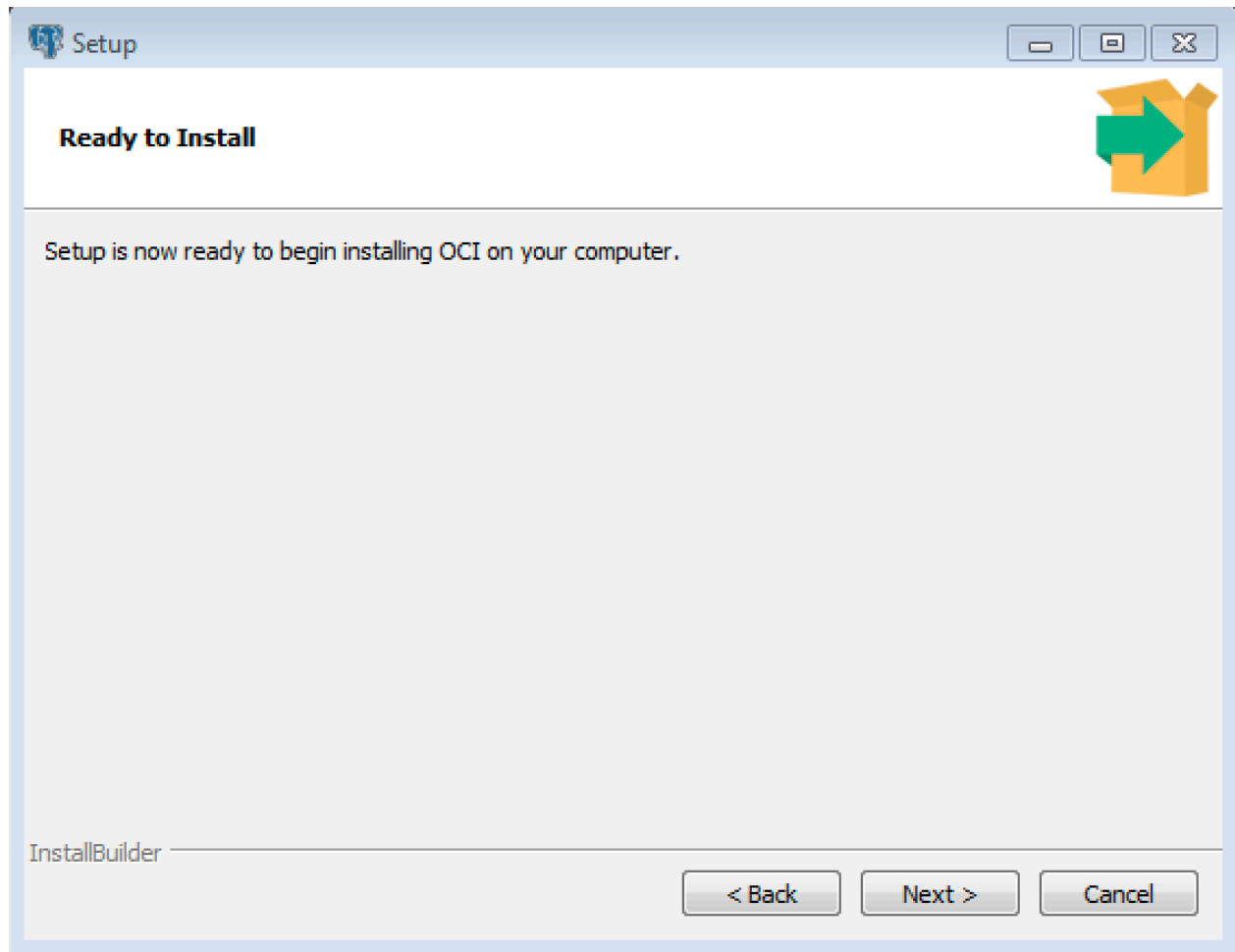


図 4 インストールの準備ダイアログ

3.2 接続文字列の形成

OCL コネクタは、Oracle スタイルと Postgres スタイルの両方の接続 URI を受け入れます。接続文字列は、次の Oracle 形式の形式をとります。

```
[//] [host] [:port] [/dbname]
```

または、以下の Postgres スタイルのフォーム:

```
postgres://[user[:password]@] [host] [:port] [/dbname]
[?param1=value1&...]
```

```
postgresql://[user[:password]@] [host] [:port] [/dbname]
[?param1=value1&...]
```

Postgres スタイルの URI を使用して、単一の URI で複数のホストコンポーネント（それぞれにオプションのポートコンポーネントを含む）を指定することもできます。マルチホスト接続文字列は次の形式を取ります。

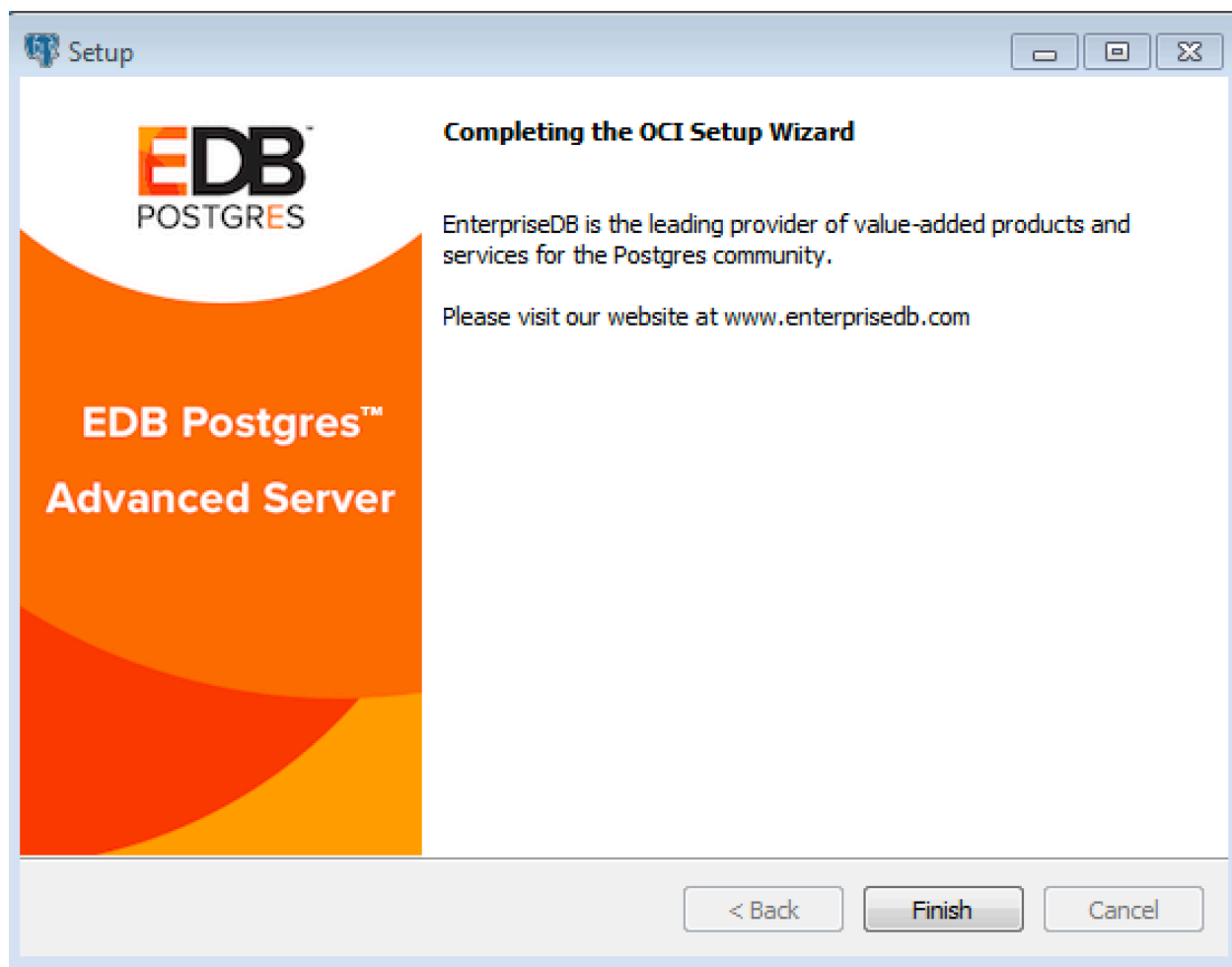


図 5 インストールが完了しました

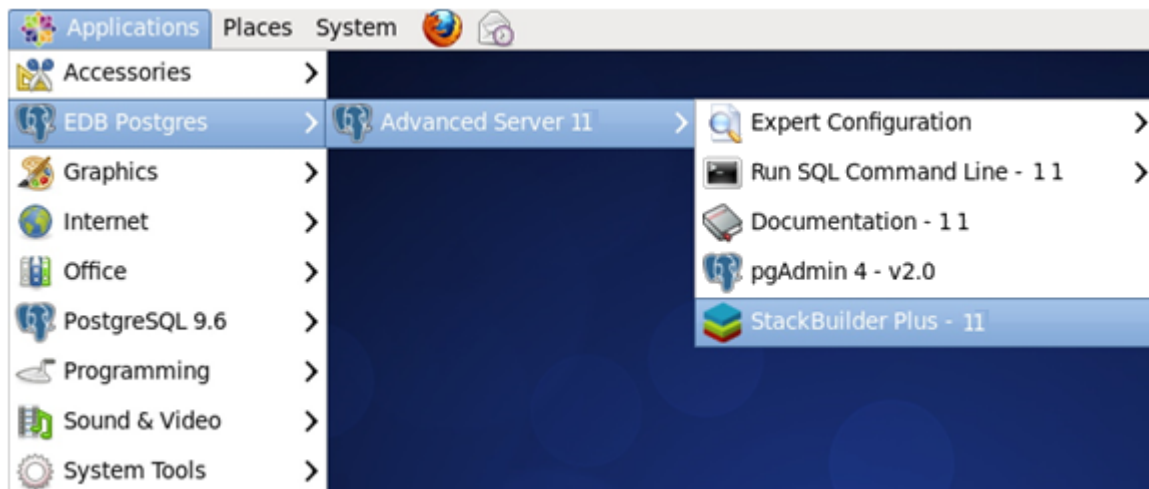


図 6 StackBuilderPlus の起動

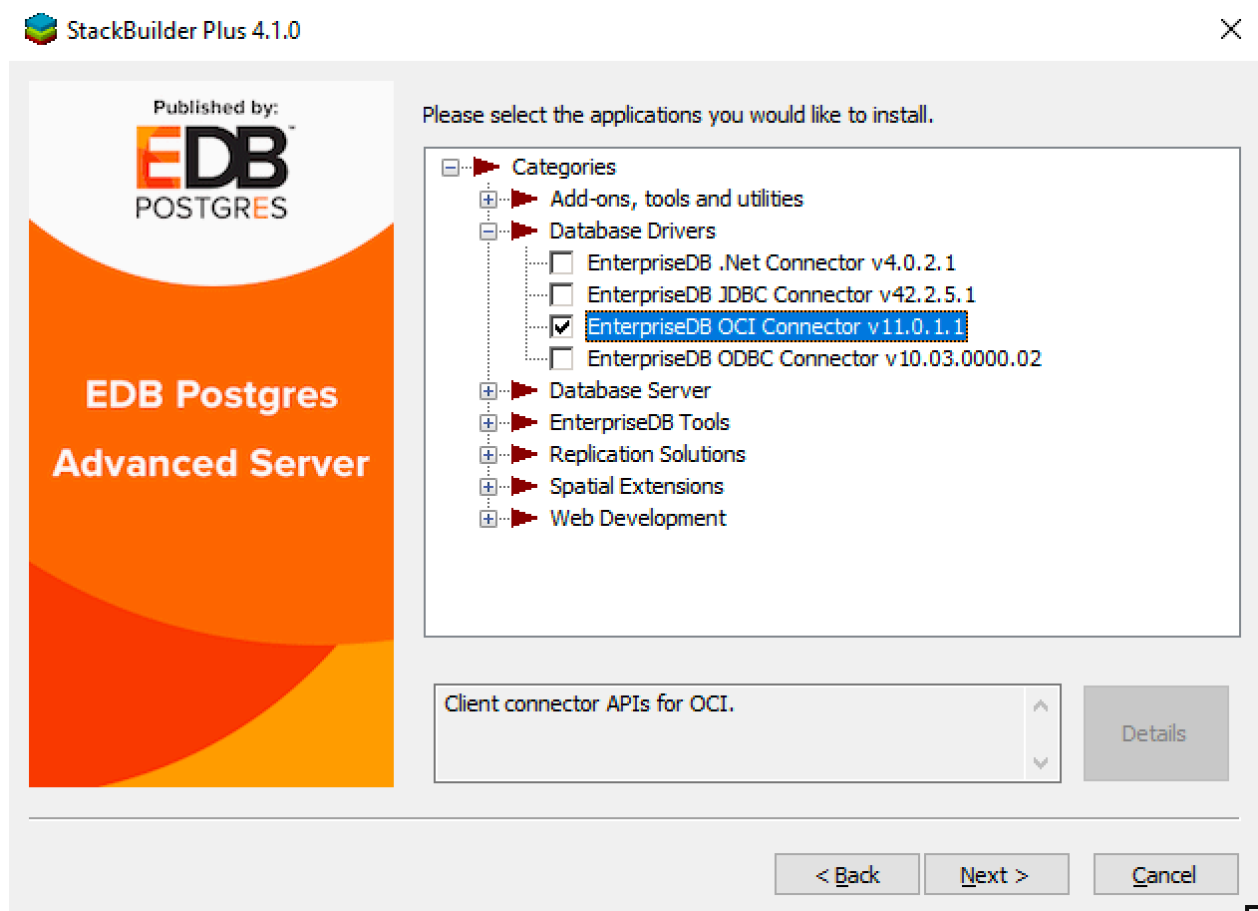


図7 コネクターインストーラーの選択

```
postgresql://host1:port1,host2:port2,host3:port3/
```

どこ:

user は接続しているユーザーの名前です。

password は接続ユーザーに関連付けられたパスワードです。

host は接続先のホスト名または IP アドレスです。IPV6 アドレスを指定するには、アドレスを角括弧で囲みます。

port は接続先のポート番号です。

dbname は接続しているデータベースの名前です。

paramx=valutex のペアは、追加の（アプリケーション固有の）接続プロパティを指定します。

たとえば、次の各接続文字列は、IP アドレスが 10.0.0.4 のシステムのポート 5444 で edb データベースへの接続を確立します。

```
//10.0.0.4:5444/edb
```

```
postgres://10.0.0.4:5444/edb
```

```
postgresql://10.0.0.4:5444/edb
```

Postgres スタイルの接続文字列の使用に関する詳細については、PostgreSQL コアのドキュメントを参照してください。<<https://www.enterprisedb.com/resources/product-documentation>>`_。

3.3 プログラムのコンパイルとリンク

EnterpriseDBOpenClientLibrary を使用すると、OracleCallInterfaceAPI を使用して記述されたアプリケーションは、C ソースコードに最小限の変更を加えるだけで、EnterpriseDB データベースに接続してアクセスできます。EnterpriseDBOpenClientLibrary ファイルの名前は次のとおりです。

Linux の場合:

```
libedboci.so
```

Windows の場合:

```
edboci.dll
```

ファイルは oci/lib サブディレクトリにインストールされます。

サンプルプログラムのコンパイルとリンク

次の例は、Linux 環境でサンプルプログラム edb_demo.c をコンパイルしてリンクします。edb_demo.c は oci/samples サブディレクトリにあります。

1. ORACLE_HOME と EDB_HOME の環境変数を設定します。

Oracle ホームディレクトリの完全なパス名に ORACLE_HOME を設定します。

例えば:

```
export ORACLE_HOME=/usr/lib/oracle/xe/app/oracle/product/10.2.0/server
```

ホームディレクトリの完全パス名に EDB_HOME を設定します。

例えば:

```
export EDB_HOME=/usr/edb
```

2. LD_LIBRARY_PATH を libpthread.so の完全なパスに設定します。デフォルトでは、libpthread.so は /lib64 にあります。

```
export LD_LIBRARY_PATH=/lib64/lib:$LD_LIBRARY_PATH
```

3. AdvancedServerOpenClient ライブラリを含めるには、LD_LIBRARY_PATH を設定します。デフォルトでは、libiconv.so.2 は \$EDB_HOME/oci/lib にあります。

```
export
```

```
LD_LIBRARY_PATH=$EDB_HOME/oci:$EDB_HOME/oci/lib:$LD_LIBRARY_PATH
```

4. 次に、OCLAPI プログラムをコンパイルしてリンクします。

```
cd $EDB_HOME/oci/samples
```

```
make
```

3.4 RefCursorSupport

AdvancedServerOpenClientLibrary は、Oracle と互換性のある PL/SQL プロシージャの OUT パラメータとして REF CURSOR の使用をサポートしています。次の API を通じてサポートが提供されます。

- OCIBindByName
- OCIBindByPos
- OCIBindDynamic
- OCISmtPrepare
- OCISmtExecute
- OCISmtFetch
- OCIAttrGet

OCL コネクタは SQLT_RSET データ型もサポートしています。

The following example demonstrates how to invoke a stored procedure that opens a cursor and returns a REF CURSOR as an output parameter. The code sample assumes that a PL/SQL procedure named openCursor (with an OUT parameter of type REF CURSOR) has been created on the database server, and that the required handles have been allocated:

```
char * openCursor =
    "begin \
      openCursor(:cmdRefCursor); \
    end;";
OCISmt *stmtOpenRefCursor;
OCISmt *stmtUseRefCursor;
```

REF CURSOR を開いて使用するストアードプロシージャを実行するためのハンドルを割り当てます:

```
/* Handle for the stored procedure to open the ref cursor */
OCIHandleAlloc((dvoid *) envhp,
               (dvoid **) &stmtOpenRefCursor,
               OCI_HTYPE_STMT,
               0,
               (dvoid **) NULL));
```

```
/* Handle for using the Ref Cursor */
OCIHandleAlloc((dvoid *) envhp,
               (dvoid **) &stmtUseRefCursor,
               OCI_HTYPE_STMT,
               0,
               (dvoid **) NULL));
```

次に、REF CURSOR を開くために使用する PL/SQL ブロックを準備します:

```
OCIStmtPrepare(stmtOpenRefCursor,
               errhp,
               (text *) openCursor,
               (ub4) strlen(openCursor),
               OCI_NTV_SYNTAX,
               OCI_DEFAULT));
```

PL/SQL openCursor OUT パラメータをバインドします:

```
OCIBindByPos(stmtOpenRefCursor,
             &bndplrcl,
             errhp,
             1,
             (dvoid *) &stmtUseRefCursor,
             /* the returned ref cursor */
             0,
             SQLT_RSET,
             /* SQLT_RSET type representing cursor */
             (dvoid *) 0,
             (ub2 *) 0,
             (ub2) 0,
             (ub4) 0,
             (ub4 *) 0,
             OCI_DEFAULT));
```

stmtOpenRefCursor ステートメントハンドルを使用して openCursor プロシージャを呼び出します:

```
OCIStmtExecute(svchp,
               stmtOpenRefCursor,
               errhp,
               1,
               0,
               0,
               0,
               OCI_DEFAULT);
```

この時点で、stmtUseRefCursor ステートメントハンドルにはカーソルへの参照が含まれています。情報を取得するには、参照カーソルの出力変数を定義します。

```
/* Define the output variables for the ref cursor */
OCIDefineByPos(stmtUseRefCursor,
               &defnEmpNo,
               errhp,
               (ub4) 1,
               (dvoid *) &empNo,
               (sb4) sizeof(empNo),
```

(次のページに続く)

(前のページからの続き)

```

        SOLT_INT,
        (dvoid *) 0,
        (ub2 *) 0,
        (ub2 *) 0,
        (ub4) OCI_DEFAULT));

```

次に、結果セットの最初の行をターゲット変数にフェッチします。

```

/* Fetch the cursor data */
OCIStmtFetch(stmtUseRefCursor,
             errhp,
             (ub4) 1,
             (ub4) OCI_FETCH_NEXT,
             (ub4) OCI_DEFAULT));

```

3.5 OCL 関数リファレンス

次の表に、OCL コネクタでサポートされる機能を示します。ヘッダーファイルはすべてユーザーが提供する必要がありますことに注意してください。AdvancedServer はそのようなファイルを提供しません。

3.5.1 接続、承認、初期化機能

関数	説明
OCIBreak	指定された OCL 関数を中止します。
OCIEnvCreate	OCL 環境を作成します。
OCIEnvInit	OCL 環境ハンドルを初期化します。
OCIInitialize	OCL 環境を初期化します。
OCILogoff	セッションを解放します。
OCILogon	ログオン接続を作成します。
OCILogon2	さまざまなモードでログオンセッションを作成します。
OCIReset	現在の操作/プロトコルをリセットします。
OCIServerAttach	データソースへのアクセスパスを確立します。
OCIServerDetach	データソースへのアクセスを削除します。
OCISessionBegin	ユーザーセッションを作成します。
OCISessionEnd	ユーザーセッションを終了します。
OCISessionGet	セッションプールからセッションを取得します。
OCISessionRelease	セッションを解放します。
OCITerminate	共有メモリサブシステムから切り離します。

tnsnames.ora ファイルの使用

OCIServerAttach メソッドは tnsnames.ora ファイルの dblink パラメータで指定された接続記述子を使用します。tnsnames.ora ファイル（Oracle データベースと互換性があります）を使用して、データベース接続アドレスを指定します。AdvancedServer は、ユーザーのホームディレクトリで tnsnames.ora という名前のファイルを検索します。AdvancedServer がユーザーのホームディレクトリで tnsnames.ora ファイルを見つけない場合、TNS_ADMIN で指定されたパスを検索します。

サンプルの tnsnames.ora ファイルには以下が含まれます:

```
EDBX =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT = 5444))
  (CONNECT_DATA = (SERVER = DEDICATED) (SID = edb))
)
```

ファイルに含まれていないパラメーターは、OpenClientLibrary によって無視されます。この例では、SID は、ポート 5444 の localhost で実行されているクラスター内の edb という名前のデータベースを指します。

tnsnames.ora ファイルを使用する OCI Server Attach への C プログラム呼び出しは次のようになります:

```
static text *username = (text *) "enterprisedb";
static text *password = (text *) "edb";
static text *attach_str = "EDBX";
OCI Server Attach(srvhp, errhp, attach_str, strlen(attach_str), 0);
```

tnsnames.ora ファイルがない場合は、接続文字列を //localhost:5444/edb の形式で指定します。

注釈: tnsnames.ora では複数の記述子もサポートされています。

3.5.2 ハンドルおよび記述子関数

関数	説明
OCIAttrGet	ハンドル属性を取得します。拡張サーバーは、次のハンドル属性をサポートしています:OCI_ATTR_USERNAME、OCI_ATTR_PASSWORD、OCI_ATTR_SERVER、OCI_ATTR_ENV、OCI_ATTR_SESSION、OCI_ATTR_ROW_COUNT、OCI_ATTR_CHARSET_FORM、OCI_ATTR_CHARSET_ID、EDB_ATTR_MODT_ATTR_STMT_LEVEL
OCIAttrSet	ハンドル属性を設定します。高度なサーバーは、次のハンドル属性をサポートしています:OCI_ATTR_USERNAME、OCI_ATTR_PASSWORD、OCI_ATTR_SERVER、OCI_ATTR_ENV、OCI_ATTR_SESSION、OCI_ATTR_ROW_COUNT、OCI_ATTR_CHARSET_FORM、OCI_ATTR_CHARSET_ID、EDB_ATTR_MODR_ATTR_MODT_ATTR_MODT_ATTR_STMT_LEVEL
OCIDescriptorPlace	記述子を割り当てて初期化します。
OCIDescriptorFree	割り当てられた記述子を解放します。
OCIHandleAlloc	ハンドルを割り当てて初期化します。
OCIHandleFree	割り当てられたハンドルを解放します。
OCIParamGet	パラメータ記述子を取得します。
OCIParamSet	パラメータ記述子を設定します。

EDB_ATTR_EMPTY_STRINGS

By default, Advanced Server will treat an empty string as a NULL value. You can use the EDB_ATTR_EMPTY_STRINGS environment attribute to control the behavior of the OCL connector when mapping empty strings. To modify the mapping behavior, use the OCIAttrSet() function to set EDB_ATTR_EMPTY_STRINGS to one of the following:

値	説明
OCI_DEFAULT	空の文字列を NULL 値として扱います。
EDB_EMPTY_STRINGS_NULL	空の文字列を NULL 値として扱います。
EDB_EMPTY_STRINGS_EMPTY	空の文字列を長さ 0 の文字列として扱います。

EDB_ATTR_EMPTY_STRINGS の値を見つけるには、OCIAttrGet () をクエリします。

EDB_ATTR_HOLDABLE

Advanced Server supports statements that execute as WITH HOLD cursors. The EDB_ATTR_HOLDABLE attribute specifies which statements execute as WITH HOLD cursors. The EDB_ATTR_HOLDABLE attribute can be set to any of the following three values:

- EDB_WITH_HOLD - WITH HOLD カーソルとして実行
- EDB_WITHOUT_HOLD - プロトコルレベルの準備済みステートメントを使用して実行します
- OCI_DEFAULT - 次の定義を参照

属性は OCISstmt ハンドルまたは OCIServer ハンドルに設定できます。OCIServer ハンドルまたは OCISstmt ハンドルを作成すると、そのハンドルの EDB_ATTR_HOLDABLE 属性は OCI_DEFAULT に設定されます。

ハンドルの EDB_ATTR_HOLDABLE 属性を変更するには OCIAttrSet () を呼び出し、属性を取得するには OCIAttrGet () を呼び出します。

AdvancedServer が SELECT ステートメントを実行するとき、それは OCIServer ハンドルの EDB_ATTR_HOLDABLE 属性を調べます。その属性が EDB_WITH_HOLD に設定されている場合、クエリは WITH HOLD カーソルとして実行されます。

OCIServer ハンドルの EDB_ATTR_HOLDABLE 属性が EDB_WITHOUT_HOLD に設定されている場合、クエリは通常の準備済みステートメントとして実行されます。

OCIServer ハンドルの EDB_ATTR_HOLDABLE 属性が OCI_DEFAULT に設定されている場合、Advanced-Server は OCIServer ハンドルの EDB_ATTR_HOLDABLE 属性の値を使用します (OCIServer の `0` 属性が EDB_WITH_HOLD に設定されている場合、クエリは ``8``カーソルとして実行されます。それ以外の場合、クエリはプロトコルで準備されたステートメントとして実行されます)。

EDB_HOLD_CURSOR_ACTION

EDB_HOLD_CURSOR_ACTION 属性は OCL インターフェースを使用して WITH HOLD カーソルが作成される方法を変更します。この属性は、次のいずれかの値に設定できます。

- EDB_COMMIT_AFTER_CURSOR - カーソルの作成後にトランザクションをコミットします
- EDB_CURSOR_WITHOUT_XACT_BLK - 新しいトランザクションチェーンを開始しない
- OCI_DEFAULT - 次の定義を参照

次に属性値について説明します。

OCI_DEFAULT

ステートメントを実行するたびに、OCL はデータベースサーバーのトランザクション状態を調べます。トランザクションがまだ進行中でない場合、OCL は **BEGIN** ステートメントを実行して新しいトランザクションブロックを作成してから、指定したステートメントを実行します。トランザクションブロックは、OCITransCommit() または OCITransRollback() を呼び出すまで開いたままです。

デフォルトでは、データベースサーバーは、コミットまたはロールバックすると、開いているカーソルをすべて閉じます。ユーザー（または OCL）が WITH HOLD 句を含むカーソルを宣言すると、カーソルの結果セットはデータベースサーバーに保持され、そのカーソルから引き続きフェッチできます。ただし、トランザクションをロールバックすると、データベースサーバーは開いているカーソルを保持しません。ROLLBACK の後にカーソルからフェッチしようとする、データベースサーバーはエラーを報告します。

EDB_COMMIT_AFTER_CURSOR

アプリケーションがトランザクションをロールバックした後に WITH HOLD カーソルから読み取る必要がある場合は、作成前に EDB_HOLD_CURSOR_ACTION を EDB_COMMIT_AFTER_CURSOR に設定することで、OCL がカーソルを作成した直後にトランザクションをコミットするように調整できますそのようなカーソル。例えば:

```
ub4          action = EDB_COMMIT_AFTER_CURSOR;

OCIAttrSet(stmt, OCI_HTYPE_STMT, &action, sizeof(action),
    EDB_ATTR_HOLD_CURSOR_ACTION, err);

OCIStmtExecute( ... );
```

EDB_COMMIT_AFTER_CURSOR を使用すると、保留中の変更がコミットされることを理解することが重要です。

EDB_CURSOR_WITHOUT_XACT_BLK

EDB_COMMIT_AFTER_CURSOR によって追加された追加のコミットでアプリケーションが適切に実行されない場合は、EDB_ATTR_HOLD_CURSOR_ACTION を EDB_CURSOR_WITHOUT_XACT_BLK に設定してみてください。このアクションでは、OCL は新しいトランザクションチェーンを開始しません。トランザクションをコミットまたはロールバックした直後に WITH HOLD カーソルを作成すると、カーソルは独自のトランザクション内に作成され、データベースサーバーはそのトランザクションをコミットし、カーソルは存続します。

カーソル宣言がトランザクション内の最初のステートメントでない場合でもエラーが発生する可能性があることを理解することが重要です。カーソルを宣言する前に他のステートメントを実行すると、WITH HOLD カーソルがトランザクションブロックに作成され、エラーが発生した場合（またはアプリケーションが OCITransRollback() を呼び出した場合）はロールバックされることがあります。

Please note that you can set the EDB_HOLD_CURSOR_ACTION on the server level (OCI Server) or for each statement handle (OCI Stmt). If the statement attribute is set to a value other than OCI_DEFAULT, the value is derived from the statement handle, otherwise (if the statement attribute is set to OCI_DEFAULT), the value is taken from the server handle. So you can define a server-wide default action by setting the attribute in the server handle, and leaving the attribute set to OCI_DEFAULT in the statement handles. You can use different values for each statement handle (or server handle) as you see fit.

EDB_ATTR_STMT_LVL_TX

特に指示がない限り、OCL コネクタは、サーバーがエラーを報告するたびに現在のトランザクションを ROLLBACK にします。選択した場合、自動 ROLLBACK を `edb_stmt_level_tx` パラメーターでオーバーライドできます。これにより、1 つ（または複数）のステートメントがトランザクション内でエラーを発生させた場合でも、トランザクション内の変更が保持されます。

You can use the `OCIServer` attribute with `OCIAttrSet()` and `OCIAttrGet()` to enable or disable `EDB_ATTR_STMT_LEVEL_TX`. By default, `edb_stmt_level_tx` is disabled. To enable `edb_stmt_level_tx`, the client application must call `OCIAttrSet()`:

```
OCIServer *server = myServer;
ub1        enabled = 1;

OCIAttrSet(server, OCI_HTYPE_SERVER, &enabled,
            sizeof(enabled), EDB_ATTR_STMT_LEVEL_TX, err);
```

`edb_stmt_level_tx` を無効にするには:

```
OCIServer *server = myServer;
ub1        enabled = 0;

OCIAttrSet(server, OCI_HTYPE_SERVER, &enabled,
            sizeof(enabled), EDB_ATTR_STMT_LEVEL_TX, err);
```

3.5.3 関数のバインド、定義、説明

関数	説明
OCIBindByName	名前でバインドします。
OCIBindByPos	位置によってバインドします。
OCIBindDynamic	バインド後に追加の属性を設定します。
OCIBindArrayOfStruct	一括操作のために構造体の配列をバインドします。
OCIDefineArrayOfStruct	配列の属性を指定します。
OCIDefineByPos	出力変数の関連付けを定義します。
OCIDefineDynamic	定義する追加の属性を設定します。
OCIDescribeAny	既存のスキーマオブジェクトについて説明します。
OCIStmtGetBindInfo	バインド変数とインジケータ変数の名前とハンドルを取得します。
OCIUserCallbackRegister	ユーザー定義のコールバックを定義します。

3.5.4 ステートメント関数

関数	説明
OCIStmtExecute	準備された SQL ステートメントを実行します。
OCIStmtFetch	データの行をフェッチします（非推奨）。
OCIStmtFetch2	データの行をフェッチします。
OCIStmtPrepare	SQL ステートメントを準備します。
OCIStmtPrepare2	SQL ステートメントを準備します。
OCIStmtRelease	ステートメントハンドルを解放します。

3.5.5 トランザクション関数

関数	説明
OCITransCommit	トランザクションをコミットします。
OCITransRollback	トランザクションをロールバックします。

3.5.6 XA 関数

関数	説明
xaoEnv	OCL 環境ハンドルを返します。
xaoSvcCtx	OCL サービスコンテキストを返します。

xaoSvcCtx

xaoSvcCtx 関数を使用するには、次のように xaoSvcCtx または xa_open 接続文字列形式の拡張を提供する必要があります:

```
Oracle_XA{+<required_fields> ...}
```

ここで、required_fields は次のとおりです:

HostName=host_ip_address は AdvancedServer データベースの IP アドレスを指定します。

PortNumber=host_port_number は AdvancedServer が実行されているポート番号を指定します。

SqlNet=dbname はデータベース名を指定します。

Acc=P/username/password はデータベースのユーザー名とパスワードを指定します。***password***は省略できます。その場合、フィールドは Acc=P/username/ として指定されます。

AppName=app_id はアプリケーションを識別する番号を指定します。

接続文字列の例を次に示します。

```
Oracle_XA+HostName=192.168.1.1+PortNumber=1533+SqlNet=XE+Acc=P/user/
↳password+AppName=1234
```

3.5.7 日付および日時関数

関数	説明
OCIDateAddDays	日数を加算または減算します。
OCIDateAddMonths	月数を加算または減算します。
OCIDateAssign	日付を割り当てます。
OCIDateCheck	指定された日付が有効かどうかを確認してください。
OCIDateCompare	2 つの日付を比較します。
OCIDateDaysBetween	2 つの日付間の日数を求めます。
OCIDateFromText	文字列を日付に変換します。
OCIDateGetDate	日付の日付部分を取得します。
OCIDateGetTime	日付の時刻部分を取得します。
OCIDateLastDay	月の最終日の日付を取得します。
OCIDateNextDay	翌日の日付を取得します。
OCIDateSetDate	日付の日付部分を設定します。
OCIDateSetTime	日付の時刻部分を設定します。
OCIDateSysDate	現在のシステムの日付と時刻を取得します。
OCIDateToText	日付を文字列に変換します。
OCIDateTimeAssign	日時の割り当てを実行します。
OCIDateTimeCheck	日付が有効かどうかを確認してください。
OCIDateTimeCompare	2 つの日時値を比較します。
OCIDateTimeConstruct	日時記述子を作成します。
OCIDateTimeConvert	ある日時タイプを別の日時タイプに変換します。
OCIDateTimeFromArray	サイズ OCI_DT_ARRAYLEN の配列を OCIDateTime 記述子に変換します。
OCIDateTimeFromText	指定された文字列を、指定された形式に従って OCIDateTime 記述子の Oracle 日時型に変換します。
OCIDateTimeGetDate	日時値の日付部分を取得します。
OCIDateTimeGetTime	日時値の時間部分を取得します。
OCIDateTimeGetTimeZoneName	日時値のタイムゾーン名の部分を取得します。
OCIDateTimeGetTimeZoneOffset	日時値のタイムゾーン（時間、分）部分を取得します。
OCIDateTimeSubtract	2 つの日時値を入力として取り、それらの差を間隔として返します。
OCIDateTimeSysTimeStamp	システムの現在の日付と時刻をタイムゾーン付きのタイムスタンプとして取得します。
OCIDateTimeToArray	OCIDateTime 記述子を配列に変換します。
OCIDateTimeToText	指定された日付を、指定された形式に従って文字列に変換します。

3.5.8 インターバル関数

関数	説明
OCIIntervalAdd	2 つの間隔値を加算します。
OCIIntervalAssign	1 つの間隔値を別の間隔値にコピーします。
OCIIntervalCompare	2 つの間隔値を比較します。
OCIIntervalGetDaySecond	間隔から日、時間、分、秒、秒の端数を抽出します。
OCIIntervalSetDaySecond	間隔内の日、時間、分、秒、および端数秒を変更します。
OCIIntervalGetYearMonth	間隔から年と月の値を抽出します。
OCIIntervalSetYearMonth	間隔で年と月の値を変更します。
OCIIntervalDivide	OCINumber 値による OCIInterval 値の除算を実装します。
OCIIntervalMultiply	OCINumber 値による OCIInterval 値の乗算を実装します。
OCI インターバル減算	別の間隔値から 1 つの間隔値を減算します。
OCIIntervalToText	間隔から文字列を外挿します。
OCIIntervalCheck	間隔値の有効性を検証します。
OCIIntervalToNumber	OCIInterval 値を OCINumber 値に変換します。
OCIIntervalFromNumber	OCINumber 値を OCIInterval 値に変換します。
OCIDateTimeIntervalAdd	OCIDatetime 値に OCIInterval 値を追加して、OCIDatetime 値を生成します。
OCIDateTimeIntervalSub	OCIDatetime 値から OCIInterval 値を減算し、OCIDatetime 値を生成します。
OCIIntervalFromText	テキスト文字列を間隔に変換します。
OCIIntervalFromTZ	時間帯指定を間隔値に変換します。

3.5.9 数値関数

関数	説明
OCINumberAbs	絶対値を計算します。
OCINumberAdd	NUMBER を追加します。
OCINumberArcCos	逆余弦を計算します。
OCINumberArcSin	逆正弦を計算します。
OCINumberArcTan	逆正接を計算します。
OCINumberArcTan2	2 つの NUMBER の逆正接を計算します。
OCINumberAssign	NUMBER を別の NUMBER に割り当てます。
OCINumberCeil	NUMBER の上限を計算します。
OCINumberCmp	NUMBER を比較します。
OCINumberCos	コサインを計算します。
OCINumberDec	NUMBER を減らします。
OCINumberDiv	2 つの NUMBER を割ります。
OCINumberExp	e を指定された NUMBER 乗します。
OCINumberFloor	NUMBER のフロアを計算します。
OCINumberFromInt	整数を OracleNUMBER に変換します。
OCINumberFromReal	実数を OracleNUMBER に変換します。
OCINumberFromText	文字列を OracleNUMBER に変換します。
OCINumberHypCos	双曲線余弦を計算します。
OCINumberHypSin	双曲線正弦を計算します。
OCINumberHypTan	双曲線正接を計算します。
OCINumberInc	NUMBER を増分します。
OCINumberIntPower	与えられた底を整数乗します。
OCINumberIsInt	NUMBER が整数かどうかをテストします。
OCINumberIsZero	NUMBER がゼロかどうかをテストします。
OCINumberLn	自然対数を計算します。
OCINumberLog	任意の底を対数として計算します。
OCINumberMod	分割モジュール。
OCINumberMul	NUMBER を乗算します。
OCINumberNeg	NUMBER を無効にします。
OCINumberPower	基数への累乗 e。
OCINumberPrec	NUMBER を指定した小数点以下の桁数に丸めます。
OCINumberRound	NUMBER を指定した小数点以下の桁数に丸めます。
OCINumberSetPi	NUMBER を Pi に初期化します。
OCINumberSetZero	NUMBER をゼロに初期化します。
OCINumberShift	10 を掛けて、指定した小数点以下の桁数をシフトします。

次のページに続く

表 2 – 前のページからの続き

OCINumberSign	NUMBER の符号を取得します。
OCINumberSin	サインを計算します。
OCINumberSqrt	NUMBER の平方根を計算します。
OCINumberSub	NUMBER を減算します。
OCINumberTan	接線を計算します。
OCINumberToInt	NUMBER を整数に変換します。
OCINumberToReal	NUMBER を実数に変換します。
OCINumberToRealArray	NUMBER の配列を実際の配列に変換します。
OCINumberToText	NUMBER を文字列に変換します。
OCINumberTrunc	NUMBER を指定した小数点以下で切り捨てます。

3.5.10 文字列関数

関数	説明
OCIStrAllocSize	文字列メモリの割り当てサイズをバイト単位で取得します。
OCIStrAssign	文字列に文字列を割り当てます。
OCIStrAssignText	文字列にテキスト文字列を割り当てます。
OCIStrPtr	文字列ポインタを取得します。
OCIStrResize	文字列メモリのサイズを変更します。
OCIStrSize	文字列のサイズを取得します。

3.5.11 カートリッジサービスとファイル I/O インターフェイス関数

関数	説明
OCIFileClose	開いているファイルを閉じます。
OCIFileExists	ファイルが存在するかどうかをテストして確認します。
OCIFileFlush	バッファリングされたデータをファイルに書き込みます。
OCIFileGetLength	ファイルの長さを取得します。
OCIFileInit	OCIFile パッケージを初期化します。
OCIFileOpen	ファイルを開きます。
OCIFileRead	ファイルからバッファに読み込みます。
OCIFileSeek	ファイル内の現在位置を変更します。
OCIFileTerm	OCIFile パッケージを終了します。
OCIFileWrite	ファイルに buflen バイトを書き込みます。

3.5.12 LOB 関数

関数	説明
OCILobRead	LOB 値（または LOB 値の一部）を返します。
OCILOBWriteAppend	データを LOB 値に追加します。
OCILobGetLength	LOB 値の長さを返します。
OCILobTrim	LOB 値の末尾からデータをトリミングします。
OCILobOpen	他の LOB 関数で使用するために LOB 値を開きます。
OCILobClose	LOB 値を閉じます。

3.5.13 その他の機能

関数	説明
OCIClientVersion	クライアントライブラリのバージョンを返します。
OCLErrorGet	エラーメッセージを返します。
OCIPGErrorGet	libpq またはサーバーによって報告されたネイティブエラーメッセージを返します。署名は次のとおりです。 剣 OCIPGErrorGet (dvoid/*hndlp, ub4recordno, OraText/*errcodep, ub4errbufsiz, OraText/*bufp, ub4bufsiz, ub4type)
OCIPasswordChange	パスワードを変更する。
OCIPing	接続とサーバーがアクティブであることを確認します。
OCIServerVersion	Oracle バージョン文字列を取得します。

3.5.14 サポートされているデータ型

関数	説明
ANSI_DATE	ANSI 日付
SQLT_AFC	ANSI 固定文字
SQLT_AVC	ANSI 可変文字
SQLT_BDOUBLE	バイナリーダブル
SQLT_BIN	バイナリデータ
SQLT_BFLOAT	バイナリフロート
SQLT_CHR	文字列
SQLT_DAT	Oracle の日付
SQLT_DATE	ANSI 日付
SQLT_FLT	浮く
SQLT_INT	整数
SQLT_LBI	ロングバイナリ
SQLT_LNG	長いです
SQLT_LVB	より長いバイナリ
SQLT_LVC	長い long (文字)
SQLT_NUM	Oracle 数値
SQLT_ODT	OCL 日付タイプ
SQLT_STR	ゼロで終了する文字列
SQLT_TIMESTAMP	タイムスタンプ
SQLT_TIMESTAMP_TZ	タイムゾーン付きのタイムスタンプ
SQLT_TIMESTAMP_LTZ	ローカルタイムゾーンのタイムスタンプ
SQLT_UIN	符号なし整数
SQLT_VBI	VCS 形式のバイナリ
SQLT_VCS	可変文字
SQLT_VNU	先行する長さバイトを持つ数
SQLT_VST	OCL 文字列タイプ

3.6 OCL エラーコード-リファレンス

次の表は、OCL コネクタによって定義されたエラーコードマッピングの一覧です。データベースサーバーがエラーコードまたは条件（1 列目または 2 列目に表示）を報告すると、OCL は値を 3 列目に表示される互換性のある値に変換します。

エラーコード	条件名	Oracle エラーコード
42601	構文エラー	ORA-16945
42P01	undefined_table	ORA-00942
02000	データなし	ORA-01403
08000	connection_exception	ORA-12545
08003	connection_does_not_exist	ORA-12545
08006	connection_failure	ORA-12545
08001	sqlclient_unable_to_establish_sqlconnection	ORA-12545
08004	sqlserver_rejected_establishment_of_sqlconnection	ORA-12545
25000	invalid_transaction_state	ORA-01453
08007	transaction_resolution_unknown	ORA-01453
0A000	feature_not_supported	ORA-03001
22012	Division_by_zero	ORA-01476
2200B	escape_character_conflict	ORA-01424
22019	invalid_escape_character	ORA-00911
2200D	invalid_escape_octet	ORA-01424
22025	invalid_escape_sequence	ORA-01424
22P06	nonstandard_use_of_escape_character	ORA-01424
2200C	invalid_use_of_escape_character	ORA-01424
22004	null_value_not_allowed	ORA-01400
23000	integer_constraint_violation	ORA-00001
23505	unique_violation	ORA-00001
40P01	t_r_deadlock_detected	ORA-00060
42701	duplicate_column	ORA-01430
53000	不十分なリソース	ORA-01659
53100	disk_full	ORA-01659
53200	out_of_memory	ORA-82100
42P07	duplicate_table	ORA-00955
21000	cardinality_violation	ORA-01427
22003	numeric_value_out_of_range	ORA-01426
22P02	invalid_text_representation	ORA-01858
28000	invalid_authorization_specification	ORA-01017

次のページに続く

表 3 – 前のページからの続き

エラーコード	条件名	Oracle エラーコード
28P01	無効なパスワード	ORA-01017
2200F	zero_length_character_string	ORA-01425
42704	undefined_object	ORA-01418
2BP01	dependent_objects_still_exist	ORA-02429
22027	trim_error	ORA-30001
22001	string_data_right_truncation	ORA-01401
22002	null_value_no_indicator_parameter	ORA-01405
22008	datetime_field_overflow	ORA-01800
44000	with_check_option_violation	ORA-01402
01007	warning_privilege_not_granted	ORA-00000
01006	warning_privilege_not_revoked	ORA-00000
02001	no_additional_dynamic_result_sets_returned	ORA-00000
03000	sql_statement_not_yet_complete	ORA-00000
08P01	protocol_violation	ORA-00000
23001	restrict_violation	ORA-00000
23502	not_null_violation	ORA-00000
23505	foreign_key_violation	ORA-00000
23514	check_violation	ORA-00000
24000	invalid_cursor_state	ORA-01001
26000	invalid_sql_statement_name	ORA-00000
42830	invalid_foreign_key	ORA-00000
55006	object_in_use	ORA-00000
55P03	lock_not_available	ORA-00054
72000	snapshot_too_old	ORA-01555

Postgres エラーコードの詳細については、PostgreSQL コアのドキュメントを参照してください<<https://www.postgresql.org/docs/12/static/errcodes-appendix.html>>`_。

3.7 マルチスレッドのサポート

OCL はマルチスレッド環境でサポートされています。モードパラメータの値として OCI_THREADED を指定して OCIEnvNlsCreate() を呼び出すことにより、マルチスレッド環境でマルチスレッドを有効化/使用できます。

```
retCode = OCIEnvNlsCreate( &envp,  
                           OCI_THREADED,  
                           NULL,  
                           NULL,  
                           NULL,  
                           NULL,  
                           0,  
                           NULL,  
                           0,  
                           0 );
```

以降の OCIEnvNlsCreate() への呼び出しもすべて OCI_THREADED で行う必要があります。

マルチスレッドアプリケーションがスレッドセーフオペレーティングシステムで実行されている場合、OCI ライブラリは、環境ハンドルごとにアプリケーションのミューテックスを管理します。

第 4 章

OCL トレースの生成

OCL トレースオプションは、指定された OCI_DEBUG_LOG file のバックエンドとの直接通信（クエリ、更新など）をログに記録します。さらに、呼び出された関数/API も記録します。トレースファイルは、デフォルトの作業ディレクトリ（oci_log_file_name）に生成されます。パスにファイル名（directory path/oci_log_file_name）を追加すると、トレースファイルが特定の場所に生成されます。

トレースファイルは、接続ごとにテキストファイル（読み取り可能）形式で生成されます。

注釈: OCL トレースはデフォルトで無効になっています。

OCL トレースを生成するには:

1. OCL の EDB クライアント側トレースを有効にします。以下の環境変数を設定することにより、OCL トレースを有効にすることができます。

```
export OCI_DEBUG_LEVEL=4
```

```
export OCI_DEBUG_LOG=oci_log_file
```

2. 環境変数をエクスポートしたら、アプリケーションを実行します。指定したディレクトリに OCL トレースファイルが生成されます。

第 5 章

セキュリティと暗号化

5.1 SSL の使用

EDBPostgresAdvancedServer は、SSL 接続を使用してクライアント/サーバー通信を暗号化し、セキュリティを強化するためのネイティブサポートを提供します。OCL では、`sslmode` パラメータを `verify-full` または `verify-ca` に設定し、検証するルート証明書をシステムに提供することで制御されます。

SSL 設定の手順:

1. サーバーおよびクライアント側の証明書を構成します。SSL クライアントおよびサーバー側の証明書の構成の詳細については、PostgreSQLSSL のドキュメントを参照してください。(<<https://www.postgresql.org/docs/12/libpq-ssl.html>>`_。
2. SSLOCL 接続を有効にします。

OCL クライアントアプリケーションでは、`Session` に `EDB_ATTR_SSL` 属性を設定することで SSL モードを有効にできます。

```
char*sslmode= "verify-full";
retValue=OCIAttrSet((dvoid*)authp, (ub4)OCI_HTYPE_SESSION,
                    (dvoid*)sslmode, (ub4)strlen((char*)sslmode),
                    (ub4)EDB_ATTR_SSL, errhp);
```

注釈: `EDB_ATTR_SSL` は、インストールディレクトリにある `edboci.h` ヘッダーファイルで定義されています。

3. SSL 属性を設定した後、`OCILogon` 関数を使用して接続を作成できます:

```
OCILogon(pEnv, pError, &pSvc, (OraText*)pUsername, (ub4)UsernameLen,
          (OraText*)pPassword, (ub4>PasswordLen,
          (OraText*)pDatabase, (ub4)DatabaseLen);
```

サーバーが認証されると、クライアントは機密データを渡す準備ができます。

サポートされている SSL モードオプションの詳細については、以下を参照してください。

<https://www.postgresql.org/docs/12/libpq-ssl.html#LIBPQ-SSL-SSLMODE-STATEMENTS>

5.2 スクラムの互換性

EDBOCL ドライバーは、AdvancedServer バージョン 11 以降に SCRAM-SHA-256 サポートを提供します。このサポートは、EDBOCL11.0.1 リリース以降で利用できます。

第 6 章

結論

EDBPostgresOCL ガイド

Copyright©2007-2019EnterpriseDBCorporation。

全著作権所有。

EnterpriseDB®Corporation

34CrosbyDrive、Suite201、Bedford、MA01730、USA

T+17813573390F+19784671307E

info@enterprisedb.com

www.enterprisedb.com

- EDB は、EDB 製品が存在する場合、ログオン UI の設計、コーディングのベストプラクティスの確立、レビュー、および検証を行います。EDB は、追加の入力コンポーネントについても同じアプローチに従いますが、製品の性質上、制限付きの検証が可能な信頼できるユーザーが入力および送信するフリーフォーム SQL、WMI、またはその他の文字列を受け入れることが必要になる場合があります。このような場合、ユーザーが誤った入力や危険な入力をするのを防ぐことはできません。
- EDB は、認証済みの信頼できるユーザーからのフリーフォーム SQL、WMI、またはその他の潜在的に危険な入力を受け入れる製品に機能を追加する権利を留保しますが、そのようなすべての機能が最小限のリスクを提供できるように設計およびテストされることを保証します。可能、スーパーユーザーまたは同等の権限が必要です。
- EDB は、すべての潜在的な脅威を予測できること、または予測できることを保証するものではありません。したがって、私たちのプロセスでは、すべての潜在的な脆弱性が対処または検討されたことを完全に保証できません。

索引

Authorize Functions, 24
 Bind Functions, 30
 Cartridge Services and File I/O Interface Functions, 36
 compiling and linking program, 20
 Connect Functions, 24
 Date and Datetime Functions, 32
 Define Functions, 30
 Describe Functions, 30
 EDB_ATTR_HOLDABLE, 27
 EDB_ATTR_STMT_LVL_TX, 28
 EDB_HOLD_CURSOR_ACTION, 27
 graphical installation, 14
 Handle and Descriptor Functions, 26
 Initialize Functions, 24
 Installation on Debian, 13
 Installation on SLES 12, 11
 Installation on Ubuntu, 13
 Interval Functions, 33
 LOB Functions, 36
 Miscellaneous Functions, 37
 Number Functions, 34
 OCL エラーコード-リファレンス, 39
 OCL コネクタのインストールと設定, 8
 OCL トレースの生成, 43
 OCL 関数リファレンス, 24
 OpenClientLibrary, 7
 Ref cursor support, 21
 RefCursorSupport, 21
 SSL の使用, 45
 Statement Functions, 30
 String Functions, 36
 Supported Data Types, 38
 Supported server versions, 5
 tnsnames.ora, 24
 Transaction Functions, 30
 XA Functions, 31
 xaoSvcCtx, 31
 サポートされているプラットフォーム, 5
 スクラムの互換性, 46

セキュリティと暗号化, 45
 プログラムのコンパイルとリンク, 20
 マルチスレッドのサポート, 41
 結論, 47
 新機能, 3
 接続文字列の形成, 16