



EDB

EDB OCL Connector

リリース **13.1.4.2**

EDB OCL Connector User's Guide

2021 年 01 月 30 日

目次

第 1 章	What' s New	3
第 2 章	Supported Platforms	5
第 3 章	libpq Cross-version Compatibility	7
第 4 章	Open Client Library	9
4.1	Installing and Configuring the OCL Connector	10
4.1.1	RPM パッケージを使用したコネクタのインストール	10
4.1.2	SLES12 ホストへのコネクタのインストール	18
4.1.3	Debian または Ubuntu ホストへのコネクタのインストール	19
4.1.4	グラフィカルインストーラーを使用してコネクタをインストールする	21
4.2	Forming a Connection String	23
4.3	Compiling and Linking a Program	26
4.4	Ref Cursor Support	28
4.5	OCL Function Reference	31
4.5.1	Functions の接続、承認、初期化	31
4.5.2	ハンドル関数と記述子 Functions	33
4.5.3	Functions のバインド、定義、および説明	37
4.5.4	ステートメント Functions	37
4.5.5	トランザクション Functions	37
4.5.6	XAFunctions	38
4.5.7	日付および日時 Functions	39
4.5.8	インターバル Functions	40
4.5.9	数字 Functions	41
4.5.10	文字列 Functions	43
4.5.11	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 Functions	43
4.5.12	LOBFunctions	43
4.5.13	その他の Functions	44
4.5.14	サポートされているデータ型	45
4.6	OCL Error Codes – Reference	46
4.7	Multithreading Support	48
4.8	OTL Support	49

4.8.1	OTL 認証	49
4.8.2	例	49
第 5 章	Generating the OCL Trace	55
第 6 章	Security and Encryption	57
6.1	Using SSL	57
6.2	Scram Compatibility	58
第 7 章	Conclusion	59
索引		61

EDBOCL コネクタは、オラクル CallInterface に似た API を提供します。オラクル CallInterface を使用するように作成されたアプリケーションは、AdvancedServer データベースサーバと対話オーダーに、EDB の OCL コネクタを使用して再コンパイルできます。

このガイドでは、以下に関するインストールおよび使用手順を説明します。

- コネクタのインストール方法。
- オラクルの接続文字列をフォームする方法。
- プログラムをコンパイルおよびリンクする方法。

このガイドには、AdvancedServer でサポートされてい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: EDB は、オラクル RealApplicationClusters (RAC) およびオラクル Exadata での OpenClientLibrary の使用をサポートしていません。前述のオラクル製品は、この EDB 製品では評価も認定もされていません。

第 1 章

What's New

EDBOCL コネクタ 13.1.4.2 を作成するために、次の機能が追加されました。

- EDBOCL は、OTL4.0 認定のサポートを提供するようになりました。詳細については、[Section-OTLSupport](#) を参照してください。
- EDBOCL インストールでは、常に最新の libpq が使用されます。詳細については、[Section-libpqCross-versionCompatibility](#) を参照してください。

第 2 章

Supported Platforms

EDBOCL コネクタは、AdvancedServer バージョン 9.5 以上で認定されています。EDBOCL コネクタネイティブ パッケージソフトは、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) 7.x および 8.x
- CentOS (x86_64) 7.x および 8.x
- OELLinux7.x および 8.x
- RHEL または CentOS7.x を実行している PPC-LE8
- SLES12.x
- Debian9.x および 10.x
- Ubuntu18.04 および 20.04LTS

EDBOCLConnector 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。

- WindowsServer2019
- WindowsServer2016
- WindowsServer2012R2
- Windows10
- Windows8.1

EDBOCL 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。

- Windows10
- Windows8.1

第 3 章

libpq Cross-version Compatibility

EDBOCL インストールでは、常に最新の libpq が使用されます。libpq のバージョン間の互換性の下でサポートされるさまざまなシナリオは次のとおりです。

- 最新の libpq がマシンにインストールされている場合、OCL はそれを使用します。
- 最新の libpq がまだインストールされていない場合、OCL はそれをインストールします。インストールされていても、古いバージョンの既存の libpq は使用しません。
- OCL バージョンをアップグレードすると、libpq も最新バージョンにアップグレードされます。

第 4 章

Open Client Library

OpenClientLibrary は、アプリケーションとオラクル CallInterface との相互運用性を提供します。以前はロックされていたアプリケーションは、EDBPostgresAdvancedServer またはオラクル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変更しません。

次の図は、OpenClientLibrary とオラクル CallInterface アプリケーションスタックを比較しています。

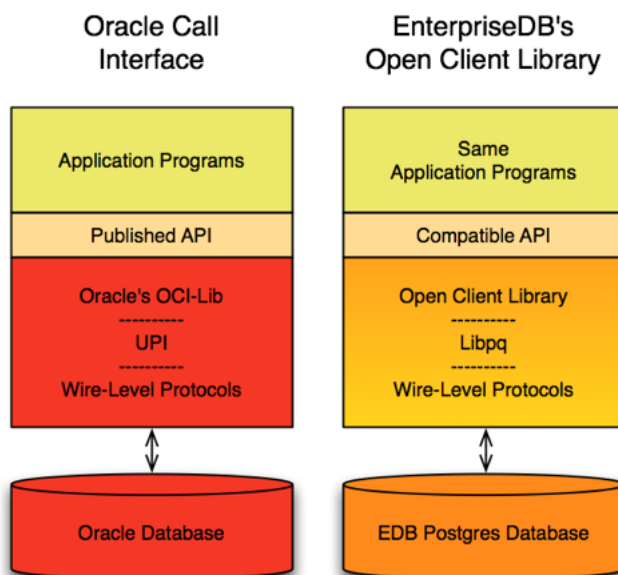


図 1 オラクル *CallInterface* との比較

OpenClientLibrary の EDB 実装は C で記述されています。

4.1 Installing and Configuring the OCL Connector

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を使用して、EDBOCL コネクタをインストールまたは更新できます。

4.1.1 RPM パッケージを使用したコネクタのインストール

次のプラットフォームで RPM パッケージを使用して OCL コネクタをインストールできます。

- *RHEL7*
- *RHEL8*
- *CentOS7*
- *CentOS8*

RHEL7 で

OCL コネクタをインストールする前に、次の前提条件パッケージソフトをインストールし、EDB から資格情報を要求する必要があります。

epel-release パッケージをインストールします。

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

オプション、エクストラ、HA リポジトリを有効にします。

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

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://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後、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. ユーザ名前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。

Creating a Repository Configuration File

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

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

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。

Modifying the file, providing your user name and password

`edb.repo` ファイルを作成した後、エディタの選択を使用して `enabled` パラメータの値が 1 であることを保証し、`baseurl` 仕様の `username` および `password` プレースホルダーを登録済み EDB ユーザの名前とパスワードに置き換えます。

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

Installing OCL Connector

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 OCL コネクタをインストールします。

```
yum install edb-oci

yum 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` は解決できない依存関係に遭遇する場合があります。存在する場合は、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。

RHEL8 で

OCL コネクタをインストールする前に、次の前提条件パッケージソフトをインストールし、EDB から資格情報を要求する必要があります。

epel-release パッケージをインストールします。

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

codeready-builder-for-rhel-8-/*-rpms リポジトリを有効にします。

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

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://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後、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. ユーザ名前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。

Creating a Repository Configuration File

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

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

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。

Modifying the file, providing your user name and password

edb.repo ファイルを作成した後、エディタの選択を使用して enabled パラメータの値が 1 であることを保証し、baseurl 仕様の username および password プレースホルダーを登録済み EDB ユーザの名前とパスワードに置き換えます。

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

Installing OCL Connector

構成ファイルへの変更を保存した後、以下のコマンドを使用して OCL コネクタをインストールします。

```
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 は解決できない依存関係に遭遇する場合があります。存在する場合は、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。

CentOS7 で

OCL コネクタをインストールする前に、次の前提条件パッケージソフトをインストールし、EDB から資格情報を要求する必要があります。

epel-release パッケージをインストールします。

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

注釈: CentOS-Base.repo ファイル (/etc/yum.repos.d にある) で [extras] リポジトリ定義を有効にする必要がある場合があります。

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://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後、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。

Creating a Repository Configuration File

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

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

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。

Modifying the file, providing your user name and password

edb.repo ファイルを作成した後、エディタの選択を使用して enabled パラメータの値が 1 であることを保証し、baseurl 仕様の username および password プレースホルダーを登録済み EDB ユーザの名前とパスワードに置き換えます。

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

Installing OCL Connector

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 OCL コネクタをインストールします。

```
yum install edb-oci  
  
yum 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 は解決できない依存関係に遭遇する場合があります。存在する場合は、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。

CentOS8 で

OCL コネクタをインストールする前に、次の前提条件パッケージソフトをインストールし、EDB から資格情報を要求する必要があります。

epel-release パッケージをインストールします。

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

PowerTools リポジトリを有効にします。

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

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

リポジトリの資格情報を受け取った後、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. ユーザ名前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。

Creating a Repository Configuration File

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

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

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。

Modifying the file, providing your user name and password

edb.repo ファイルを作成した後、エディタの選択を使用して enabled パラメータの値が 1 であることを保証し、baseurl 仕様の username および password プレースホルダーを登録済み EDB ユーザの名前とパスワードに置き換えます。

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

Installing OCL Connector

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 OCL コネクタをインストールします。

```
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 は解決できない依存関係に遭遇する場合があります。存在する場合は、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。

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、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、スーパーユーザ権限を引き受けて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 または dnf は edb.repo ファイルを更新して、edb.repo ファイルで指定された資格情報で接続するように構成された現在の EDB リポジトリにアクセスできるようにしファイル。次に、yum を使用して、インストールされているパッケージソフトをアップグレードできます。

- 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
```

4.1.2 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そうとしますが、EDB 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 EDB 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/edb-sles.repo
```

リポジトリ構成ファイルを作成した後、zypper refresh コマンドを使用して SLES ホストのメタデータをリフレッシュし、EDB リポジトリを組み込みます。

User Name および Password の入力を求められたら、EDB リポジトリの接続資格情報を入力します。リポジトリの資格情報をリクエストするには、「EDB ウェブサイト<<<https://www.enterprisedb.com/repository-access-request>>>」に<<https://www.enterprisedb.com/repository-access-request>>アクセスしてください。

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 -r 'REGISTRATION_CODE' -e 'EMAIL'
SUSEConnect -p PackageHub/12.4/x86_64
SUSEConnect -p sle-sdk/12.4/x86_64
```

SUSE ホストの登録に関する詳細については、「SUSE ウェブサイト<<<https://www.suse.com/support/kb/doc/?id=7016626>>>」に<<https://www.suse.com/support/kb/doc/?id=7016626>>アクセスしてください。

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できます。

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

4.1.3 Debian または Ubuntu ホストへのコネクタのインストール

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、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リクエストするには、「EDB ウェブサイト」https://www.enterprisedb.com/repository-access-requestに「」にアクセスしてください。

次の手順では、EDBapt 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を説明します。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。

1. スーパーユーザ特権を想定します。

```
sudo su -
```

2. EDB リポジトリを構成します。

Debian9 の場合:

```
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'
```

Debian10 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 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'
```

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. EDB 署名キーを追加しキー。

```
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
```

4.1.4 グラフィカルインストーラーを使用してコネクタをインストールする

EDB 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 EDBOCL コネクタを追加できます。ウィザードは 'EDB ウェブサイト<https://www.enterprisedb.com/software-downloads-postgres/>' で利用できます。

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。(インストーラをダウンロードし、インストーラ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。

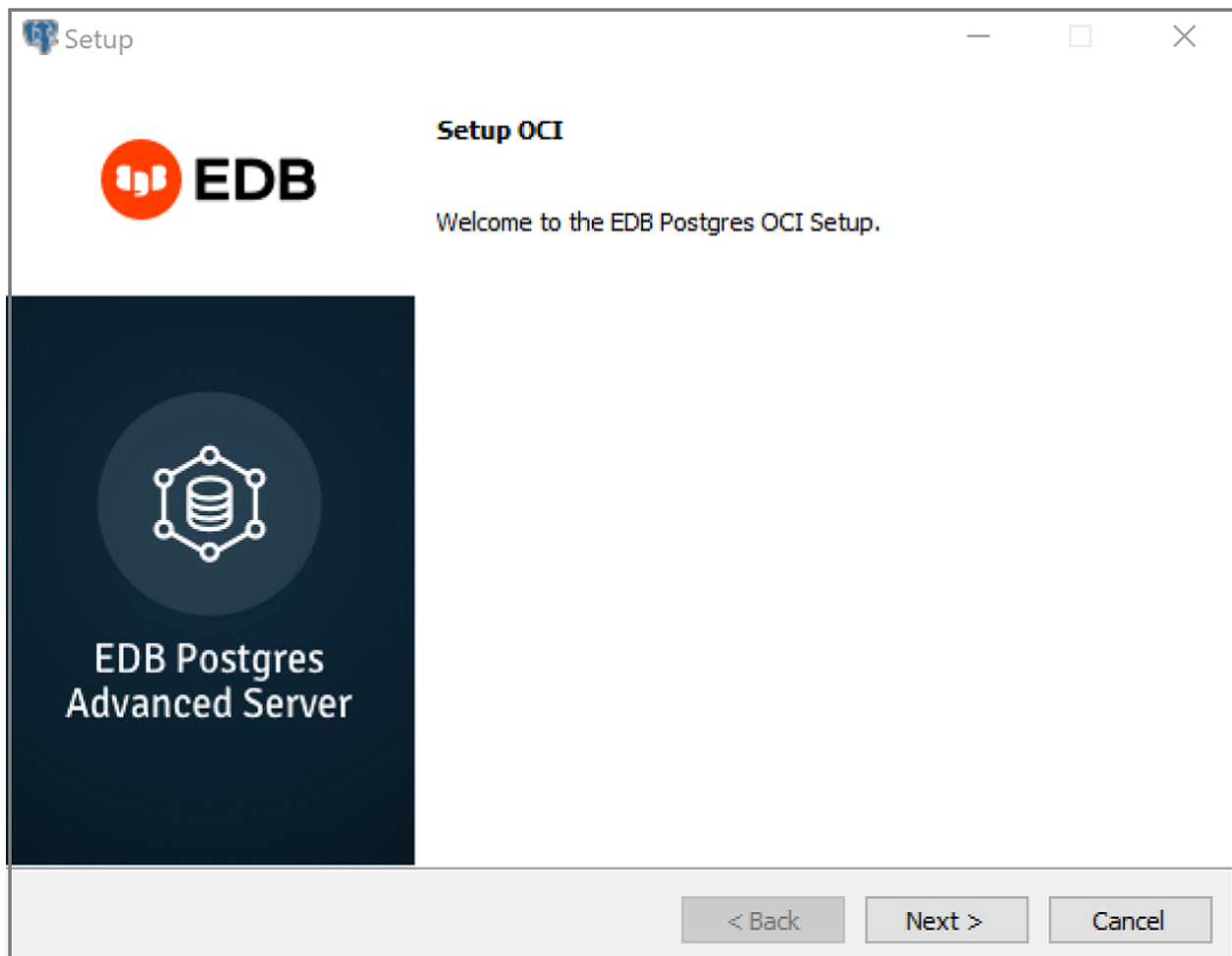


図 2 OCL コネクタインストールウィザード

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、

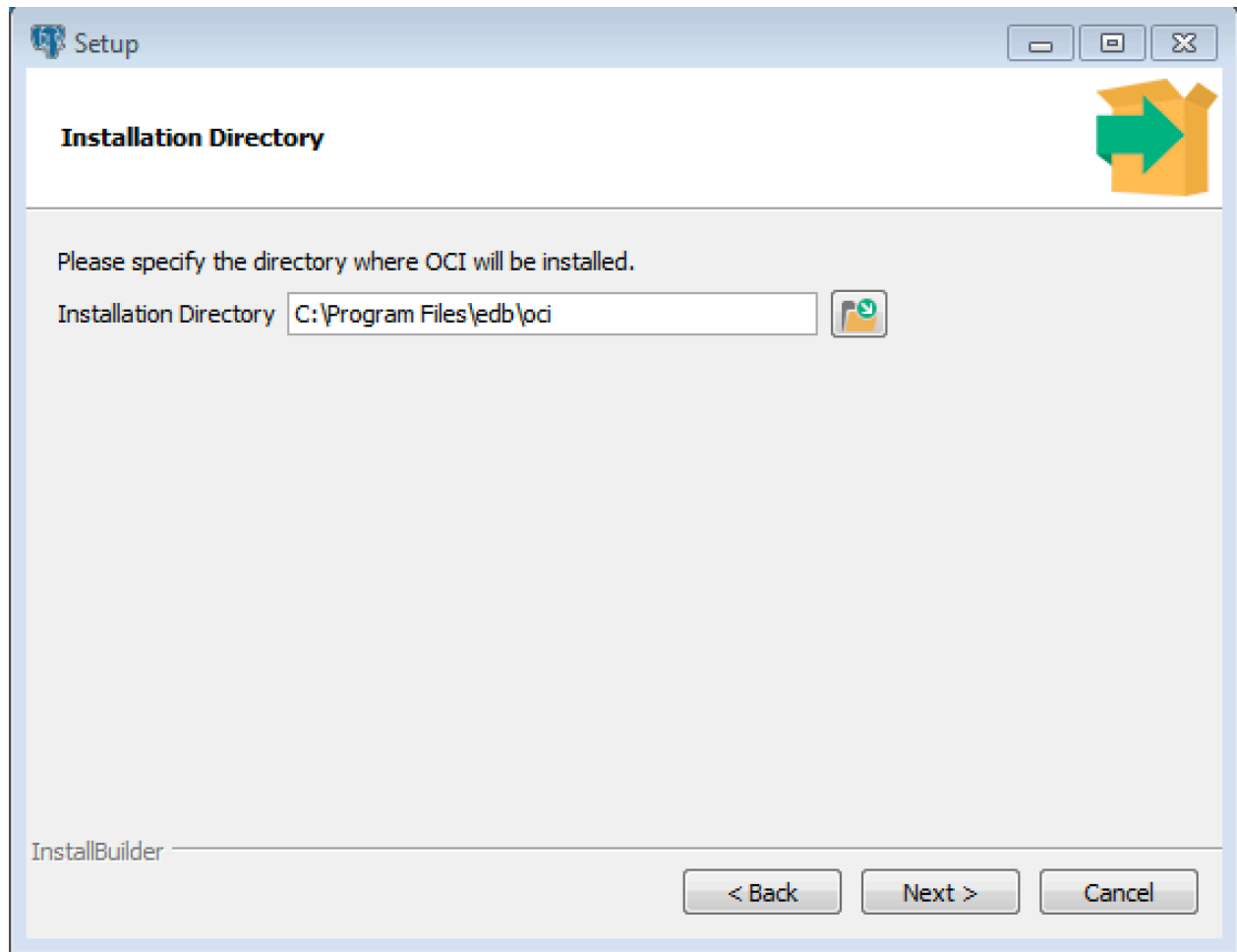


図 3 インストールダイアログ

Next をクリックして続行します。

Ready to Install ダイアログで Next をクリックして、インストールをスタートします。ポップアップダイアログがインストールウィザードの進行状況を確認します。

ウィザードがセットアップの完了を通知したら、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 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、EDB コネクタのインストールを追加または更新します。

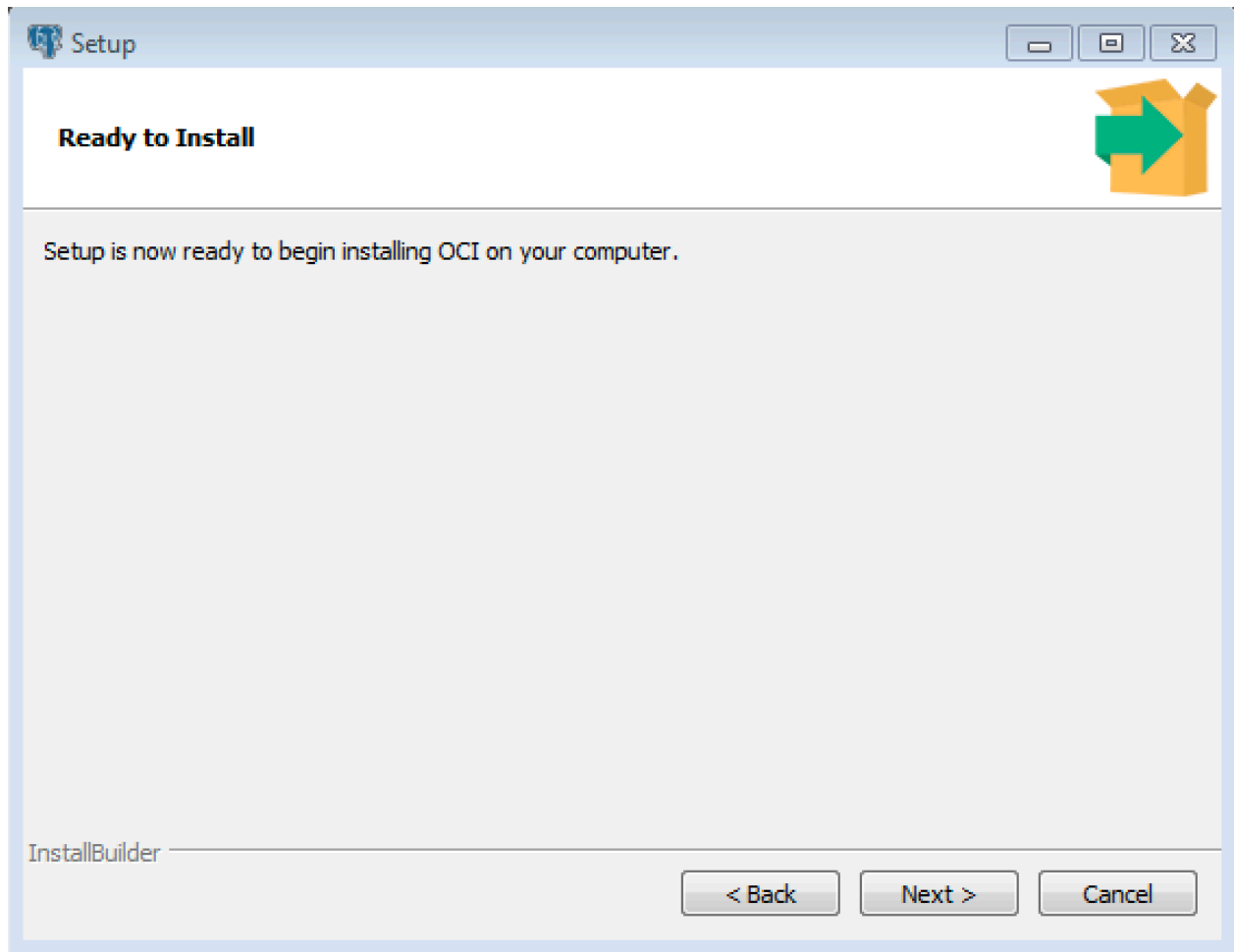


図 4 [インストールの準備完了] ダイアログ

4.2 Forming a Connection String

EDBOCL コネクターは、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 スタイルのフォームを取る場合があります。

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

または、次の 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&...]
```

また、Postgres スタイルの URI を使用して、単一の URI にマルチプルのホストコンポーネント（それぞれオプションのポートコンポーネント）を指定することもできます。マルチホスト接続文字列のフォームは次のとおり

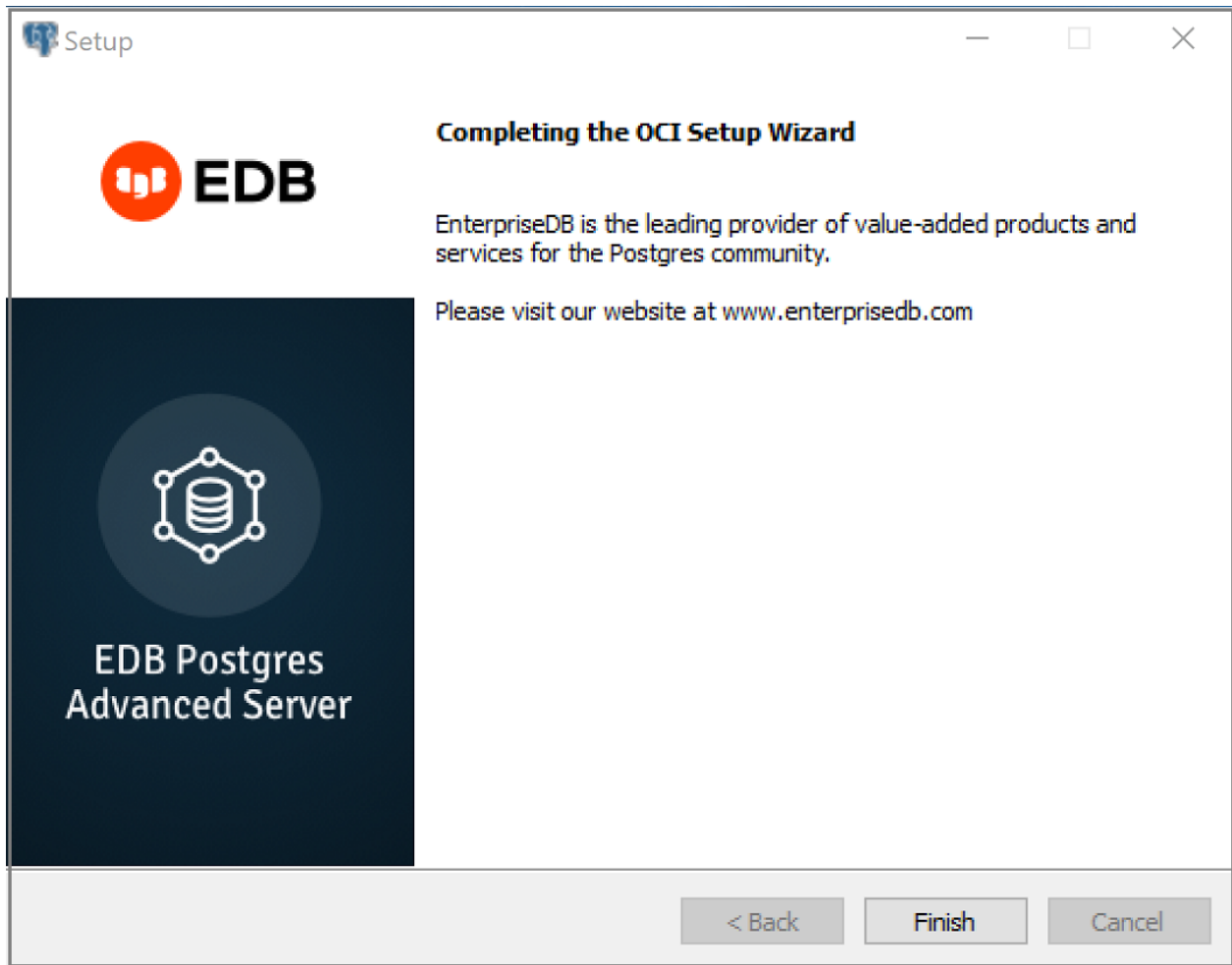


図5 インストールが完了しました

です。

```
postgresql://<user>:<password>@host1:port1,host2:port2,host3:port3/
```

どこで:

`user` は接続ユーザの名前です。

`password` は、接続しているユーザに関連付けられたパスワード。

`host` は、接続先のホスト名前または IP アドレスです。IPV6 アドレスを指定するには、アドレスを大括弧で囲みます。

`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` 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。

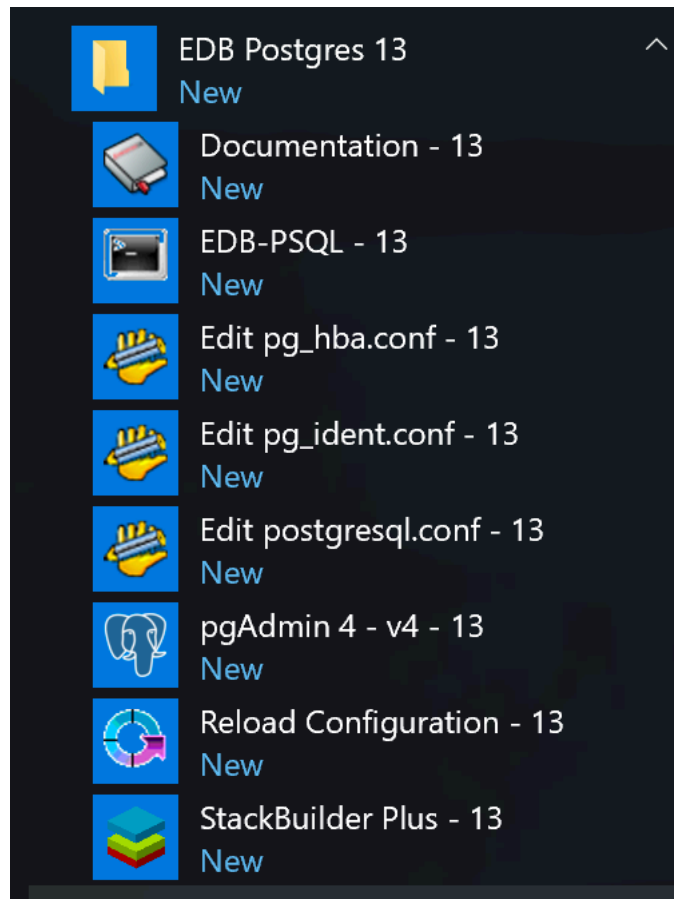


図 6 StackBuilderPlus の起動

例、次の接続文字列のそれぞれは、10.0.0.4 の IP アドレスを持つシステムのポート 5444 上 edb データベースへの接続を確立します。

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

```
postgres://<user>:<password>@10.0.0.4:5444/edb
```

```
postgresql://<user>:<password>@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、EDB ウェブサイト<<ahref="https://www.enterprisedb.com/edb-docs">https://www.enterprisedb.com/edb-docs>_にある PostgreSQL コア文書を参照してください。

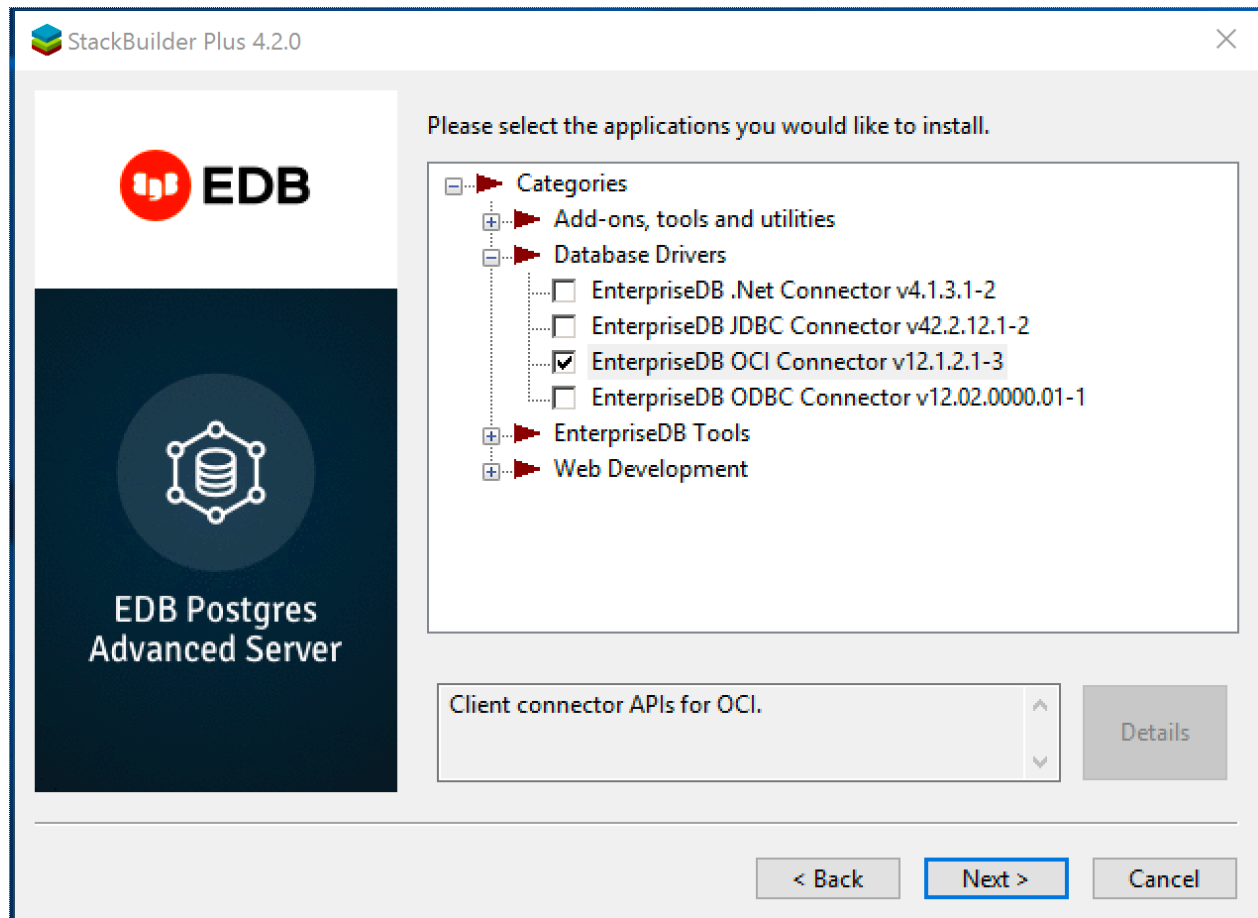


図 7 コネクタインストーラの選択

4.3 Compiling and Linking a Program

EDBOpenClientLibrary により、オラクル CallInterfaceAPI を使用して記述されたアプリケーションは、C ソースコードへの最小限の変更で EDB データベースに接続し、アクセスできます。EDBOpenClientLibrary ファイルの名前付けは次のとおりです。

Linux の場合:

`libedboci.so`

Windows:

`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。

Compiling and Linking a Sample Program

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

1. ORACLE_HOME および EDB_HOME 環境変数を設定します。

ORACLE_HOME をオラクルホームディレクトリの完全なパス名に設定します。

例:

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

EDB_HOME をホームディレクトリの完全なパス名に設定します。

例:

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

2. LD_LIBRARY_PATH を libpthread.so の完全パスに設定します。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. LD_LIBRARY_PATH を設定して、AdvancedServerOpenClient ライブラリを含めます。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
```

4.4 Ref Cursor Support

AdvancedServerOpenClientLibrary は、オラクルと互換性のある PL/SQL プロシージャで REF CURSOR を OUT パラメータとして使用することをサポートしています。サポートは、次の API を通じて提供されます。

- OCIBindByName
- OCIBindByPos
- OCIBindDynamic
- OCISmtPrepare
- OCISmtExecute
- OCISmtFetch
- OCIAAttrGet

EDBOCL コネクターは、SQLT_RSET データタイプもサポートしています。

次の例は、カーソルを開き、出力パラメータとして REF CURSOR を返すストアードプロシージャを呼び出す方法を示しています。コードサンプルでは、openCursor 名前付けの PL/SQL プロシージャ（REF CURSOR 型の OUT パラメータを使用）がデータベースサーバに作成され、必要なハンドルが割り当てられていることを前提としています。

```
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 を開くために使用される 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、カーソルへのリファレンスが含まれています。情報を取得するには、ref 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)

```
SQLT_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));
```

4.5 OCL Function Reference

以下の表は、EDBOCL コネクタがサポートしている機能のリスト。すべてのヘッダーファイルはユーザが指定する必要がある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。

4.5.1 Functions の接続、承認、初期化

ファンクション	説明
OCIBreak	指定された OCL ファンクションを中止します。
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	現在のオペレーション/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	ユーザセッションを作成します。
OCISessionEnd	ユーザ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	セッションします。
OCITerminate	共有メモリサブシステムからデタッチします。

tnsnames.ora ファイルの使用

OCIServerAttach および OCILogon 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ディスクリプタとして NET_SERVICE_NAME を使用します。tnsnames.ora ファイル（オラクルデータベースと互換性あり）を使用して、データベースコネクションの詳細を指定します。OCL は、ユーザーのホームディレクトリで .tnsnames.ora 名前付けのファイルを検索します。OCL がユーザーのホームディレクトリで .tnsnames.ora ファイルを見つけられない場合、TNS_ADMIN 環境変数で指定されたパスで tnsnames.ora を検索します。

tnsnames.ora ファイルには、複数の記述子 (NET_SERVICE_NAME) を指定できます。

サンプルの tnsnames.ora ファイルには以下が含まれます。

```
EDBX =
(DESCRIPTION =
```

(次のページに続く)

(前のページからの続き)

```
(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 名前付けのデータベースを指します。

tnsnames.ora ファイルを使用する OCIServerAttach 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";  
OCIServerAttach(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/edbx フォームで提供します。

注釈: tnsnames.ora では複数の記述子もサポートされています。

4.5.2 ハンドル関数と記述子 Functions

ファンクション	説明
OCIAttrGet	ハンドル属性を取得します。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_STMT_LEVEL_TX、OCI_ATTR_MODE
OCIAttrSet	ハンドル属性を設定します。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_STMT_LEVELTX、 OCI_ATTR_MODE__
OCIDescriptorAlloc	ディスクリプタを割り当てて初期化します。
OCIDescriptorFree	割り当てられたディスクリプタを解放します。
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	パラメータディスクリプタを取得します。
OCIParamSet	パラメータディスクリプタを設定します。

EDB_ATTR_EMPTY_STRINGS

デフォルトでは、AdvancedServer は空の文字列を NULL 値として扱います。EDB_ATTR_EMPTY_STRINGS 環境属性を使用して、空の文字列をマッピング際の OCL コネクタの動作を制御できます。マッピングの動作を変更するには、OCIAttrSet () ファンクションを使用して EDB_ATTR_EMPTY_STRINGS を次のいずれかに設定します。

価値	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	空の文字文字列を長さゼロ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 () をクエリー。

EDB_ATTR_HOLDABLE

AdvancedServer は、WITH HOLD カーソルとして実行するステートメントをサポートしています。EDB_ATTR_HOLDABLE 属性は、WITH HOLD カーソルとして実行ステートメントを指定します。EDB_ATTR_HOLDABLE 属性は、次の 3 つの値のいずれかに設定できます。

- 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プリペアドステートメントを使用して実行
- OCI_DEFAULT - 次の定義を参照

あなたは OCISmt ハンドルや OCIServer ハンドルに属性を設定することができます。あなたは OCIServer ハンドルや OCISmt ハンドルを作成すると、そのハンドル用 EDB_ATTR_HOLDABLE 属性が OCI_DEFAULT に設定されています。

OCIAttrSet () を呼び出してハンドルの EDB_ATTR_HOLDABLE 属性を変更し、OCIAttrGet () を呼び出して属性を取得できます。

AdvancedServer は SELECT ステートメントを実行するときに、OCIServer ハンドルの EDB_ATTR_HOLDABLE 属性を調べ属性。その属性が EDB_WITH_HOLD に設定されている場合、クエリーは 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、クエリーは通常のプリペアドステートメントとして実行されます。

OCIServer ハンドルで EDB_ATTR_HOLDABLE 属性が OCI_DEFAULT に設定されている場合は、AdvancedServer は OCIServer ハンドルに EDB_ATTR_HOLDABLE 属性の値を使用しています (OCIServer で EDB_ATTR_HOLDABLE 属性が EDB_WITH_HOLD に設定されている場合、クエリーの実行 WITH HOLD カーソルとして、そうでない場合は、クエリーは、プロトコルでプリペアド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 インタフェースを使用した WITH HOLD カーソルの作成方法を変更します。この属性は、次の値のいずれかに設定できます。

- 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ません
- OCI_DEFAULT -次の定義を参照

次に、属性値について説明し属性。

OCI_DEFAULT

ステートメントを実行たびに、OCL はデータベースサーバ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、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は、コミットまたはロールバックするときに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した場合、カーソルの結果セットはデータベースサーバ、そのカーソルからフェッチを続行できます。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はエラーをレポートします。

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、カーソルを作成する前に EDB_HOLD_CURSOR_ACTION を EDB_COMMIT_AFTER_CURSOR に設定することにより、カーソル作成直後に OCL がトランザクションをコミットように手配できカーソル。例:

```
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 によって追加のコミットが追加されてアプリケーションが正しく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。このアクションにより、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で作成され、データベースサーバはそのトランザクションをコミットし、カーソルは保持されます。

カーソルを宣言する前に、いくつかの他のステートメントを実行場合は、WITH HOLD カーソルがトランザクションブロック内に作成され、バック圧延することができる-カーソル宣言は、トランザクション内の最初の文でない場合、あなたはまだエラーが発生することを理解することは重要です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）。

EDB_HOLD_CURSOR_ACTION は、サーバーレベル (OCI_Srvr) または各ステートメントハンドル (OCIStmt) で設定できることにノートしてください。ステートメント属性が OCI_DEFAULT 以外の値に設定されている場合、値はステートメントハンドルから派生します。そうでない場合 (ステートメント属性が OCI_DEFAULT に設定されている場合)、値はサーバーハンドルから取得されます。だから、サーバー・ハンドルに属性を設定すると、文ハンドルに OCI_DEFAULT に属性を残すことによって、サーバー全体のデフォルトのアクションを定義することができます。必要に応じて、ステートメントハンドル (またはサーバーハンドル) ごとに異なる値を使用できます。

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 します。選択した場合、edb_stmt_level_tx パラメータして自動 ROLLBACK をオーバーライドできます。このパラメーターは、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が発生させた場合でも、トランザクション内の変更を保持します。

OCIAttrSet() および OCIAttrGet() で OCI_Srvr 属性を使用して、EDB_ATTR_STMT_LEVEL_TX を有効または無効にできます。デフォルトでは、edb_stmt_level_tx は無効になっています。edb_stmt_level_tx を有効にするには、クライアントアプリケーションは OCIAttrSet() を呼び出す必要があります。

```
OCI_Srvr *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:

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

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

4.5.3 Functions のバインド、定義、および説明

ファンクション	説明
OCIBindByName	名前でバインドします。
OCIBindByPos	位置でバインドします。
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	定義用の追加属性を設定します。
OCIDescribeAny	既存のスキーマオブジェクトを説明します。
OCIStmtGetBindInfo	バインドおよびインジケータ変数名とハンドルを取得し変数。
OCIUserCallbackRegister	ユーザー定義のコールバックを定義します。

4.5.4 ステートメント Functions

ファンクション	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	文ハンドルを解放します。

4.5.5 トランザクション Functions

ファンクション	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。

4.5.6 XAFunctions

ファンクション	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

次のように xaoSvcCtx ファンクションを使用するオーダーに、xaoSvcCtx 又は xa_open 文字列フォーマットの拡張が提供されなければなりません。

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

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 はデータベース名前を指定します。

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。パスワードは省略できます。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。

以下は、接続文字列の例です。

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

4.5.7 日付および日時 Functions

ファンクション	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見つけます。
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	日時ディスクリプタを組み立てます。
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 OCL_DT_ARRAYLEN の配列を OCIDateTime ディスクリプタに変換します。
OCIDateTimeFromText	指定された形式に従って、指定された文字列を OCIDateTime ディスクリプタのオラクル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	日時値の時刻部分を取得します。
OCIDateTimeGetTimeZoneName	日時値のタイムゾーン名前部分を取得します。
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 datetime 値を入力として受け取り、それらの差をインターバルとして返します。
OCIDateTimeSysTimeStamp	タイムゾーン付きのタイムスタンプとしてシステムの現在の日付と時刻を取得します。
OCIDateTimeToArray	OCIDateTime ディスクリプタを配列に変換します。
OCIDateTimeToText	指定された形式に従って、指定された日付を文字列に変換します。

4.5.8 インターバル Functions

ファンクション	説明
OCIIntervalAdd	2 つのインターバル値を追加します。
OCIIntervalAssign	あるインターバル値を別のインターバル値にコピーします。
OCIIntervalCompare	2 つのインターバル値を比較します。
OCIIntervalGetDaySecond	インターバルから日、時間、分、秒、および小数秒を抽出します。
OCIIntervalSetDaySecond	インターバル内の日、時間、分、秒、および小数秒を変更します。
OCIIntervalGetYearMonth	インターバルから年と月の値を抽出します。
OCIIntervalSetYearMonth	インターバル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	OCIInterval 値と OCINumber 値の乗算を実装します。
OCIIntervalSubtract	別のインターバル値から 1 つのインターバル値を減算します。
OCIIntervalToText	インターバルから文字の並びを外挿します。
OCIIntervalCheck	インターバル値の妥当性を検証します。
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	OCIInterval 値を OCIDatetime 値に追加し、その結果 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	テキスト文字列をインターバルに変換します。
OCIIntervalFromTZ	タイムゾーンの指定をインターバル値に変換します。

4.5.9 数字 Functions

ファンクション	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	1 つの 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 をデクリメントします。
OCINumberDiv	2 つの NUMBER を分割します。
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	整数をオラクル NUMBER に変換します。
OCINumberFromReal	実数をオラクル NUMBER に変換します。
OCINumberFromText	文字列をオラクル NUMBER 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 をインクリメントします。
OCINumberIntPower	指定された基数を整数の累乗にします。
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	任意の底の対数を計算します。
OCINumberMod	モジュロ除算。
OCINumberMul	NUMBER を掛けます。
OCINumberNeg	NUMBER を無効にします。
OCINumberPower	ベース e へのべき乗。
OCINumberPrec	NUMBER を指定された 10 進数数に丸めます。
OCINumberRound	NUMBER を指定された 10 進数に丸めます。
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 で乗算し、指定された 10 進数数をシフトします。

[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 のルートを計算します。
OCINumberSub	NUMBER を引きます。
OCINumberTan	タンジェントを計算します。
OCINumberToInt	NUMBER を整数に変換します。
OCINumberToReal	NUMBER を実数に変換します。
OCINumberToRealArray	NUMBER の配列を実数配列に変換します。
OCINumberToText	NUMBER を文字列に変換します。
OCINumberTrunc	指定した 10 進数で NUMBER を切り捨てます。

4.5.10 文字列 Functions

ファンクション	説明
OCIStringAllocSize	割り当てられた文字列メモリのサイズをバイト単位で取得します。
OCIStringAssign	文字列に割り当て文字列。
OCIStringAssignText	文字列に割り当てテキスト文字列。
OCIStringPtr	文字列ポインタを取得します。
OCIStringResize	文字列メモリのサイズを変更します。
OCIStringSize	文字列サイズを取得します。

4.5.11 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 Functions

ファンクション	説明
OCIFileClose	オープン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	ファイル内の現在の位置を変更します。
OCIFileTerm	OCIFile パッケージを終了します。
OCIFileWrite	buflen バイトをファイルに書き込みます。

4.5.12 LOBFunctions

ファンクション	説明
OCILobRead	LOB 値（または LOB 値の一部）を返します。
OCILOBWriteAppend	LOB 値にデータを追加します。
OCILobGetLength	LOB 値の長さを返します。
OCILobTrim	LOB 値の末尾からデータを切り取ります。
OCILobOpen	他の LOB 関数が使用する LOB 値を開きます。
OCILobClose	LOB 値を閉じます。

4.5.13 その他の Functions

ファンクション	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。
OCIErrorGet	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。シグネチャは次のとおりです。 剣 OCIPGErrorGet(dvoid/hndlp, ub4recordno, OraText/*errcodep, ub4errbufsiz, OraText/ bufp, ub4bufsiz, ub4 タイプ)
OCIPasswordChange	パスワードを変更します。
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	オラクルバージョン文字列を取得します。

4.5.14 サポートされているデータ型

ファンクション	説明
ANSI_DATE	ANSI 日付
SQLT_AFC	ANSI 固定文字
SQLT_AVC	ANSI 変数文字
SQLT_BDOUBLE	バイナリダブル
SQLT_BIN	バイナリデータ
SQLT_BFLOAT	バイナリフロート
SQLT_CHR	文字文字列
SQLT_DAT	オラクルの日付
SQLT_DATE	ANSI 日付
SQLT_FLT	フロート
SQLT_INT	整数
SQLT_LBI	ロングバイナリ
SQLT_LNG	長い
SQLT_LVB	長いロングバイナリ
SQLT_LVC	ロングロング (文字)
SQLT_NUM	オラクル数値
SQLT_ODT	OCL 日付タイプ
SQLT_STR	ゼロ終了文字列
SQLT_TIMESTAMP	タイムスタンプ
SQLT_TIMESTAMP_TZ	タイム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 フォーマットのバイナリ
SQLT_VCS	可変文字
SQLT_VNU	先行する長さのバイトを持つ数字
SQLT_VST	OCL 文字列タイプ

4.6 OCL Error Codes – Reference

次の表に、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を示します。データベースサーバがエラーコードまたは状態（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	オラクルエラーコード
42601	syntax_error	ORA-16945
42P01	undefined_table	ORA-00942
02000	no_data	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	integrity_constraint_violation	ORA-00001
23505	unique_violation	ORA-00001
40P01	t_r_deadlock_detected	ORA-00060
42701	duplicate_column	ORA-01430
53000	insufficient_resources	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	オラクルエラーコード
28P01	invalid_password	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 ‘文書’<<ahref="https://www.postgresql.org/docs/12/static/errcodes-appendix.html">https://www.postgresql.org/docs/12/静的/errcodes-appendix.html>’を参照してください。

4.7 Multithreading Support

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() 呼び出しを行うことにより、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 );
```

OCIEnvNlsCreate() への後続の呼び出しもすべて OCI_THREADED を使用して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、各環境ハンドルのアプリケーションのミューテックスを管理します。

4.8 OTL Support

OTL (Oracle TemplateLibrary) は、データベースアクセス用の C++ ライブラリです。単一のヘッダファイルで構成されます。OTL の詳細については、次をご覧ください。

<http://otl.sourceforge.net/>

4.8.1 OTL 認証

EDBOCL コネクタ、バージョン 13.1.4.2 は OTL4.0 で認定されています。OTL がサポートするデータ型を使用し、他の OTL 固有の動作のために、OTL ベースのアプリを実行する前に、シェルで OTL 環境変数を定義する必要があります。OTL の値は重要ではなく、定義だけです。例、OTL に関連しているシナリオの条件付き実行のため OTL=TRUE をエクスポートことができます。

EDBOCL コネクタは、次の OTL 機能で認定されています。

- `otl_connect` を使用して、接続、切断、コミット、およびロールバックします。
- 静的ファンクション `otl_cursor::direct_exec` を使用した定数 SQL 文 (バインド変数がない場合、SQL 文は定数です)。CREATE TABLE や CREATE PROCEDURE/FUNCTION などのほとんどの DDL ステートメントが含まれます。
- `otl_stream class` を使用したバインド変数を持つ SQL ステートメント。SELECT、UPDATE、DELETE、INSERT、および PROCEDURE/FUNCTION 呼び出しのようなほとんどの DML ステートメントが含まれます。
- `otl_datetime` を使用した日付/時刻データ型。
- `otl_long_string` を使用した Raw/LongRaw データ型。
- `otl_refcur_stream` を使用してカーソルを参照します。

4.8.2 例

接続してログイン

次のコードは、OCL を初期化し、`tnsnames.ora` ベースの接続文字列を使用してデータベースに接続する方法を示しています。

```
otl_connect db;
otl_connect::otl_initialize();

db.rlogon("enterprisedb/edb@EDBX");
if(db.connected)
    cout<<"Connected to Database"<<endl;
```

CREATETABLE、INSERT、および SELECT

次のコードは、`otl_cursor::direct_exec` を使用してテーブルを作成し、このテーブルに行を挿入する方法を示しています。その後、`otl_stream` を使用して、挿入された行を取得できます。

```
char* createstmt = "create table testtable(c1 VARCHAR2(15), c2 DATE)";
char* insertstmt = "insert into testtable values('test_data123', TO_DATE('2005-12-31_
↳23:59:59','YYYY-MM-DD HH24:MI:SS'))";
char* selectstmt = "select c1, c2 from testtable";

otl_cursor::direct_exec(db, createstmt);    // create table
db.commit();

otl_cursor::direct_exec(db, insertstmt);    //Insert data.

char strData[100];
otl_datetime dtData;
otl_stream otlCur(50, sqlstmt,db);
while (!otlCur.eof())
{
    otlCur >> strData >> dtData;

    cout<<"Retrieved Value: "<<data<<endl;
    cout<<"Retrieved Value: "<<data.month<<"/"<<data.day<<"/"<<data.year<<" "<<data.hour<<
↳":"<<data.minute<<":"<<data.second<<endl;
}
```

更新

次のコードは、`UPDATE` ステートメントでバインドパラメータを使用する方法を示しています。

```
char* updatestmt = "UPDATE testtable SET c1=:c1<char[49]> WHERE c1=:c2<char[49]>";

char whereValue[50] = "test_data123";
char data[50] = "otl test";
otl_stream otlCur(80, updatestmt, db);
otlCur.set_commit(0);
otlCur<<data<<whereValue;
```

ストアドプロシージャ

次のコードは、`otl_cursor::direct_exec` を使用してストアドプロシージャを作成し、`otl_stream` を使用してそれ呼び出す方法を示しています。

```
otl_cursor::direct_exec
(
```

(次のページに続く)

(前のページからの続き)

```

db,
"CREATE OR REPLACE PROCEDURE my_procOneIntOut "
"  (A IN NUMBER, B OUT NUMBER) "
"IS "
"BEGIN "
"  B := A;"
"END;"
);

otl_stream otlCur(1, "begin my_procOneIntOut (:A<int,in>, :B<int,out>);end;", db);
otlCur.set_commit(0);

int a = 10;
otlCur<<a;

int b;
otlCur>>b;
cout << "B: " << b << endl;

```

ファンクション

次のコードは、`otl_cursor::direct_exec` を使用して関数を作成し、`otl_stream` を使用してファンクションを呼び出す方法を示しています。

注釈: この例では、`edb` サンプルデータベースの `emp` テーブルを使用しています。

```

otl_cursor::direct_exec
(
db,
"CREATE OR REPLACE FUNCTION get_no_int(e_name character varying(10)) "
"RETURNS int AS $$ "
"DECLARE retval int; "

"BEGIN "
"      \"SELECT empno FROM emp WHERE ename = e_name INTO retval; \"
"      \"RETURN retval; \"
"END; \"

\"$$ LANGUAGE plpgsql;\"
);

char ename[50] = \"SCOTT\";
otl_stream otlCur(1,

```

(次のページに続く)

(前のページからの続き)

```

"begin "
" :rc<int,out> := get_no_int(:cl<char[11],in>);"
"end;"
, db);
otlCur << ename;

int eno;
otlCur >> eno;

cout<<"Retrieved Value: "<<eno<<endl;

```

REFCURSOR

次のコードは、OUT パラメーターとして 3 つの REFCURSOR を返し、それを呼び出すプロシージャを使用してパッケージを作成する方法を示しています。

注釈: この例では、edb サンプルデータベースの emp テーブルを使用しています。

```

otl_cursor::direct_exec
(
db,
"CREATE OR REPLACE PACKAGE ref_test
IS
TYPE p_cursor IS REF CURSOR;
PROCEDURE getdata(empc OUT p_cursor, salc OUT p_cursor, comc OUT p_cursor);
END ref_test;"
);

otl_cursor::direct_exec
(
db,
"CREATE OR REPLACE PACKAGE BODY ref_test \
IS \
PROCEDURE getdata(empc OUT p_cursor, salc OUT p_cursor, comc OUT p_
↪cursor) IS \
BEGIN \
open empc for select empno, ename from EMP; \
open salc for select ename, sal from EMP; \
open comc for select ename, comm from EMP; \
END; \
END ref_test;"
);

otl_stream otlCur(1,

```

(次のページに続く)

(前のページからの続き)

```

        "BEGIN \
        ref_test.getdata(:cur1<refcur,out[50]>, :cur2<refcur,out[50]>, :cur3
↪<refcur,out[50]>); \
        END;",
        db
    );
otlCur.set_commit(0);

otl_refcur_stream s1; // reference cursor streams for reading rows.
otl_refcur_stream s2; // reference cursor streams for reading rows.
otl_refcur_stream s3; // reference cursor streams for reading rows.

otlCur>>s1;
otlCur>>s2;
otlCur>>s3;

int e_no;
char name[11];
double sal;
double comm;

cout<<"====> Reading :cur1..."<<endl;
while(!s1.eof()){ // while not end-of-data
    s1>>e_no>>name;
    cout <<"e_no=" <<e_no <<"\tname: " << name <<endl;
}

cout<<"====> Reading :cur2..."<<endl;
while(!s2.eof()){ // while not end-of-data
    s2>>name>>sal;
    cout <<"name=" <<name <<"\tsalary: " << sal <<endl;
}

cout<<"====> Reading :cur3..."<<endl;
while(!s3.eof()){ // while not end-of-data
    s3>>name>>comm;
    cout <<"name=" <<name <<"\tcommission: " << comm <<endl;
}

s1.close();
s2.close();
s3.close();

```


第 5 章

Generating the OCL Trace

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。パスにファイル名前（directory path/oci_log_file_name）を追加すると、トレースファイルは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）形式で生成されフォーマット。

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。OCL トレースファイルは、指定されたディレクトリに生成されます。

第 6 章

Security and Encryption

6.1 Using SSL

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、セキュリティする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。

Steps of SSL configuration:

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・サーバ側の証明書の設定の詳細については、PostgreSQLSSL 文書<<<https://www.postgresql.org/docs/12/libpq-ssl.html>>>を参照して<<<https://www.postgresql.org/docs/12/libpq-ssl.html>>>ください。

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 の `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、インストレーションディレクトリにある `edboci.h` ヘッダファイルで定義されています。

3. SSL 属性を設定した後、`OCILogon` ファンクション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整います。

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/13/libpq-ssl.html#LIBPQ-SSL-SSLMODE-STATEMENTS>

6.2 Scram Compatibility

EDBOCL ドライバは、AdvancedServer バージョン 11 以降の 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可能です。

第 7 章

Conclusion

EDBOCL コネクタガイド

Copyright©2007-2021EnterpriseDBCorporation。

禁無断転載

EnterpriseDB

34CrosbyDrive、Suite201、Bedford、MA01730、アメリカ

T+17813573390F+19784671307E

info@enterprisedb.com

www.enterprisedb.com

- EnterpriseDB および PostgresEnterpriseManager は、EnterpriseDBCorporation の登録商標です。EDB および EDBPostgres は、EnterpriseDBCorporation の商標です。オラクルは、オラクル Inc の登録商標です。その他の商標は、それぞれの所有者の商標です。
- EDB は、存在する場合、EDB 製品のログオン 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、および入力バリデーションを行います。EDB は追加の入力コンポーネントに対して同じアプローチに従いますが、製品の性質上、限定されたバリデーションが可能な信頼できるユーザーによって入力および送信されるフリーフォーム SQL、WMI またはその他の文字列を受け入れる必要がある場合があります。このような場合、ユーザーが誤った入力や危険な入力を禁止することはできません。
- EDB は、フリーフォーム SQL または将来的に認証され、信頼されたユーザーからの他の潜在的に危険な入力を受け入れるの製品に機能を追加する権利を留保しますが、彼らは最低限のリスクの可能性を提供確保する保証に設計され、テストされているようなすべての機能を保証し、どこ可能であれば、スーパーユーザまたは同等の特権が必要です。
- EDB は、すべての潜在的な脅威を予測できることを保証しません。したがって、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プロセスが完全に保証することはできません。

索引

Authorize Functions,31 Bind Functions,37 Cartridge Services and File I/O Interface Functions,43 centos7,14 centos8,16 Compiling and Linking a Program,26 compiling and linking program,26 Conclusion,59 Connect Functions,31 Date and Datetime Functions,39 Define Functions,37 Describe Functions,37 EDB_ATTR_HOLDABLE,34 EDB_ATTR_STMT_LVL_TX,36 EDB_HOLD_CURSOR_ACTION,34 Forming a Connection String,23 Generating the OCL Trace,55 graphical installation,21 Handle and Descriptor Functions,33 Initialize Functions,31 Installation on Debian,19 Installation on SLES 12,18 Installation on Ubuntu,19 Installing and Configuring the OCL Connector,10 Interval Functions,40 libpq Cross-version Compatibility,7 LOB Functions,43 Miscellaneous Functions,44 Multithreading Support,48 Number Functions,41 OCL Error Codes - Reference,46 OCL Function Reference,31 Open Client Library,9 OTL Support,49 Ref Cursor Support,28 Ref cursor support,28 rhel8,12 Scram Compatibility,58 Security and Encryption,57 Statement Functions,37 String Functions,43 Supported Data Types,45	Supported Platforms,5 Supported server versions,5 tnsnames.ora,31 Transaction Functions,37 Using SSL,57 What' s New,3 XA Functions,38 xaoSvcCtx,38
---	--