Contents

1 What's New

2 Supported Platforms

3 Open Client Library
   3.1 Installing and Configuring the OCL Connector ........................................ 6
      3.1.1 Installing the Connector with an RPM Package ...................................... 6
          On RHEL 7 ................................................................................. 6
          On RHEL 8 ................................................................................. 8
          On CentOS 7 ............................................................................. 10
          On CentOS 8 ............................................................................. 12
          Updating an RPM Installation ................................................................ 13
      3.1.2 Installing the Connector on an SLES 12 Host ........................................ 14
      3.1.3 Installing the Connector on a Debian or Ubuntu Host .............................. 15
      3.1.4 Using the Graphical Installer to Install the Connector ............................ 17
   3.2 Forming a Connection String ................................................................. 23
   3.3 Compiling and Linking a Program ............................................................ 24
   3.4 Ref Cursor Support ............................................................................. 25
   3.5 OCL Function Reference ....................................................................... 28
      3.5.1 Connect, Authorize and Initialize Functions ......................................... 28
          Using the tnsnames.ora File ................................................................ 28
      3.5.2 Handle and Descriptor Functions ......................................................... 30
          EDB_ATTR_EMPTY_STRINGS ................................................................ 31
          EDB_ATTR_HOLDABLE ..................................................................... 31
          EDB_HOLD_CURSOR_ACTION ................................................................ 32
          EDB_ATTR_STMT_LVL_TX .................................................................. 33
      3.5.3 Bind, Define and Describe Functions ................................................... 34
      3.5.4 Statement Functions ....................................................................... 34
      3.5.5 Transaction Functions ..................................................................... 34


<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.6</td>
<td>XA Functions</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>xaoSvcCtx</td>
<td>35</td>
</tr>
<tr>
<td>3.5.7</td>
<td>Date and Datetime Functions</td>
<td>36</td>
</tr>
<tr>
<td>3.5.8</td>
<td>Interval Functions</td>
<td>37</td>
</tr>
<tr>
<td>3.5.9</td>
<td>Number Functions</td>
<td>38</td>
</tr>
<tr>
<td>3.5.10</td>
<td>String Functions</td>
<td>40</td>
</tr>
<tr>
<td>3.5.11</td>
<td>Cartridge Services and File I/O Interface Functions</td>
<td>40</td>
</tr>
<tr>
<td>3.5.12</td>
<td>LOB Functions</td>
<td>40</td>
</tr>
<tr>
<td>3.5.13</td>
<td>Miscellaneous Functions</td>
<td>41</td>
</tr>
<tr>
<td>3.5.14</td>
<td>Supported Data Types</td>
<td>42</td>
</tr>
<tr>
<td>3.6</td>
<td>OCL Error Codes – Reference</td>
<td>43</td>
</tr>
<tr>
<td>3.7</td>
<td>Multithreading Support</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Generating the OCL Trace</td>
<td>46</td>
</tr>
<tr>
<td>5</td>
<td>Security and Encryption</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Using SSL</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>Scram Compatibility</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>Index</td>
<td>51</td>
</tr>
</tbody>
</table>
The EDB OCL Connector provides an API similar to the Oracle Call Interface. Applications that are written to use the Oracle Call Interface may be recompiled using EDB’s OCL connector in order to interact with an Advanced Server database server.

This guide provides installation and usage instructions about:

- How to install the connector.
- How to form an Oracle style connection string.
- How to compile and link a program.

This guide also includes a reference section for the functions supported by Advanced Server.

Note: EDB does not support use of the Open Client Library with Oracle Real Application Clusters (RAC) and Oracle Exadata; the aforementioned Oracle products have not been evaluated nor certified with this EDB product.
CHAPTER 1

What’s New

The following features are added to create the EDB OCL Connector 13.1.4.1:

- Support for EDB Postgres Advanced Server 13.
- Support for Ubuntu 20.04 LTS platform.
The EDB OCL Connector is certified with Advanced Server version 9.5 and above. The EDB OCL Connector native packages are supported on the following 64-bit platforms:

- Red Hat Enterprise Linux (x86_64) 7.x and 8.x
- CentOS (x86_64) 7.x and 8.x
- OEL Linux 7.x and 8.x
- PPC-LE 8 running RHEL or CentOS 7.x
- SLES 12.x
- Debian 9.x and 10.x
- Ubuntu 18.04 and 20.04 LTS

The EDB OCL Connector graphical installers are supported on the following 64-bit Windows platforms:

- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2
- Windows 10
- Windows 8.1

The EDB OCL Connector graphical installers are supported on the following 32-bit Windows platforms:
• Windows 10
• Windows 8.1
The Open Client Library provides application interoperability with the Oracle Call Interface - an application that was formerly locked in can now work with either an EDB Postgres Advanced Server or an Oracle database with minimal to no changes to the application code.

The following diagram compares the Open Client Library and Oracle Call Interface application stacks.

![Comparison with Oracle Call Interface](image)

**Fig. 1:** *Comparison with Oracle Call Interface*
The EDB implementation of the Open Client Library is written in C.

3.1 Installing and Configuring the OCL Connector

You can use an RPM package, a native package, or a graphical installer to install or update the EDB OCL Connector.

3.1.1 Installing the Connector with an RPM Package

You can install the OCL Connector using an RPM package on the following platforms:

- RHEL 7
- RHEL 8
- CentOS 7
- CentOS 8

On RHEL 7

Before installing the OCL Connector, you must install the following prerequisite packages, and request credentials from EDB:

Install the epel-release package:

```
```

Enable the optional, extras, and HA repositories:

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

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

```
```

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-oci`.

### Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

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

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

### Modifying the file, providing your user name and password

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

```ini
[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

After saving your changes to the configuration file, use the following commands to install the OCL Connector:

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

When you install an RPM package that is signed by a source that is not recognized by your system, `yum` may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, `yum` may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.
On RHEL 8

Before installing the OCL Connector, you must install the following prerequisite packages, and request credentials from EDB:

Install the `epel-release` package:

```
```

Enable the `codeready-builder-for-rhel-8-\*--rpms` repository:

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

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:


After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-oci`.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

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

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

Modifying the file, providing your user name and password

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

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:<password>@yum.enterprisedb.com/
    edb/redhat/rhel-$releasever-$basearch
enabled=1
```

(continues on next page)
Installing OCL Connector

After saving your changes to the configuration file, use the below command to install the OCL Connector:

```
dnf install edb-oci

dnf install edb-oci-devel
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter \texttt{y}, and press \texttt{Return} to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.
On CentOS 7

Before installing the OCL Connector, you must install the following prerequisite packages, and request credentials from EDB:

Install the epel-release package:

```
```

**Note:** You may need to enable the [extras] repository definition in the CentOS-Base.repo file (located in /etc/yum.repos.d).

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

```
```

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install edb-oci.

**Creating a Repository Configuration File**

To create the repository configuration file, assume superuser privileges, and invoke the following command:

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

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

**Modifying the file, providing your user name and password**

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

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:<password>@yum.enterprisedb.com/
edb/redhat/rhel-$releasever-$basearch
enabled=1
```

(continues on next page)
Installing OCL Connector

After saving your changes to the configuration file, use the following command to install the OCL Connector:

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

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.
On CentOS 8

Before installing the OCL Connector, you must install the following prerequisite packages, and request credentials from EDB:

Install the `epel-release` package:

```
```

Enable the `PowerTools` repository:

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

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:


After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-oci`.

**Creating a Repository Configuration File**

To create the repository configuration file, assume superuser privileges, and invoke the following command:

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

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

**Modifying the file, providing your user name and password**

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

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

3.1. Installing and Configuring the OCL Connector
Installing OCL Connector

After saving your changes to the configuration file, use the following command to install the OCL Connector:

```bash
dnf install edb-oci

dnf install edb-oci-devel
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press Return to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

Updating an RPM Installation

If you have an existing OCL Connector RPM installation, you can use yum or dnf to upgrade your repository configuration file and update to a more recent product version. To update the `edb.repo` file, assume superuser privileges and enter:

- On RHEL or CentOS 7:
  ```bash
  yum upgrade edb-repo
  ```
- On RHEL or CentOS 8:
  ```bash
  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:

- On RHEL or CentOS 7:
  ```bash
  yum upgrade edb-oci
  yum upgrade edb-oci-devel
  ```
- On RHEL or CentOS 8:
  ```bash
  dnf upgrade edb-oci
  dnf upgrade edb-oci-devel
  ```
3.1.2 Installing the Connector on an SLES 12 Host

You can use the zypper package manager to install the connector on an SLES 12 host. zypper will attempt to satisfy package dependencies as it installs a package, but requires access to specific repositories that are not hosted at EDB. Before installing the connector, use the following commands to add EDB repository configuration files to your SLES host:

```
zypper addrepo
https://zypp.enterprisedb.com/suse/edb-sles.repo
```

After creating the repository configuration files, use the `zypper refresh` command to refresh the metadata on your SLES host to include the EDB repositories.

When prompted for a User Name and Password, provide your connection credentials for the EDB repository. To request credentials for the repository, visit the EDB website.

Before installing EDB Postgres Advanced Server or supporting components, you must also add SUSEConnect and the SUSE Package Hub extension to the SLES host, and register the host with SUSE, allowing access to SUSE repositories. Use the commands:

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

For detailed information about registering a SUSE host, visit the SUSE website.

Then, you can use the zypper utility to install the connector:

```
zypper install edb-oci
zypper install edb-oci-devel
```
3.1.3 Installing the Connector on a Debian or Ubuntu Host

To install a package on a Debian or Ubuntu host, you must have credentials that allow access to
the EDB repository. To request credentials for the repository, visit the EDB website.

The following steps will walk you through on using the EDB apt repository to install a Debian
package. When using the commands, replace the username and password with the credentials
provided by EDB.

1. Assume superuser privileges:

   sudo su -

2. Configure the EDB repository:

   On Debian 9:

   ```bash
   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'
   ```

   On Debian 10:

   a. Set up the EDB repository:

   ```bash
   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. Substitute your EDB credentials for the username and password in the following
   command:

   ```bash
   sh -c 'echo "machine apt.enterprisedb.com login <username>
   →password <password>" > /etc/apt/auth.conf.d/edb.conf'
   ```

3. Add support to your system for secure APT repositories:

   ```bash
   apt-get install apt-transport-https
   ```

4. Add the EDB signing key:

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

5. Update the repository metadata:

   ```bash
   apt-get update
   ```
6. Install DEB package:

```
apt-get install edb-oci
apt-get install edb-oci-dev
```
3.1.4 Using the Graphical Installer to Install the Connector

You can use the EDB Connectors Installation wizard to add the EDB OCL connector to your system; the wizard is available at the EDB website.

This section demonstrates using the Installation Wizard to install the Connectors on a Windows system. (Download the installer, and then, right-click on the installer icon, and select Run As Administrator from the context menu.)

When the Language Selection popup opens, select an installation language and click OK to continue to the Setup window.

![Fig. 2: The OCL Connector Installation wizard](image)

Click Next to continue.
Use the Installation Directory dialog to specify the directory in which the connector will be installed, and click Next to continue.
Click **Next** on the **Ready to Install** dialog to start the installation; popup dialogs confirm the progress of the installation wizard.
When the wizard informs you that it has completed the setup, click the Finish button to exit the dialog.

You can also use StackBuilder Plus to add or update the connector on an existing Advanced Server installation; to open StackBuilder Plus, select StackBuilder Plus from the Windows Apps menu.
When StackBuilder Plus opens, follow the onscreen instructions. Select the EnterpriseDB OCI Connector option from the Database Drivers node of the tree control.
Follow the directions of the onscreen wizard to add or update an installation of the EDB Connectors.
3.2 Forming a Connection String

The EDB OCL connector accepts both Oracle-style and Postgres-style connection URI’s. A connection string may take the following Oracle-style form:

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

or the following Postgres-style forms:

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

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

You can also use a Postgres-style URI to specify multiple host components (each with an optional port component) in a single URI. A multi-host connection string takes the form:

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

Where:

- **user** is the name of the connecting user.
- **password** is the password associated with the connecting user.
- **host** is the host name or IP address to which you are connecting; to specify an IPV6 address, enclose the address in square brackets.
- **port** is the port number to which you are connecting.
- **dbname** is the name of the database with which you are connecting.
- **paramx=valuex** pairs specify extra (application-specific) connection properties.

For example, each of the following connection strings establish a connection to the *edb* database on port 5444 of a system with an IP address of 10.0.0.4:

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

For more information about using Postgres-style connection strings, please see the PostgreSQL core documentation, available at the EDB website.
3.3 Compiling and Linking a Program

The EDB Open Client Library allows applications written using the Oracle Call Interface API to connect to and access an EDB database with minimal changes to the C source code. The EDB Open Client Library files are named:

On Linux:
libedboci.so

On Windows:
edboci.dll

The files are installed in the oci/lib subdirectory.

Compiling and Linking a Sample Program

The following example compiles and links the sample program edb_demo.c in a Linux environment. The edb_demo.c is located in the oci/samples subdirectory.

1. Set the ORACLE_HOME and EDB_HOME environment variables.

   Set ORACLE_HOME to the complete pathname of the Oracle home directory.
   
   For example:
   
   export ORACLE_HOME=/usr/lib/oracle/xe/app/oracle/product/10.2.0/server
   
   Set EDB_HOME to the complete pathname of the home directory.
   
   For example:
   
   export EDB_HOME=/usr/edb

2. Set LD_LIBRARY_PATH to the complete path of libpthread.so. By default, libpthread.so is located in /lib64.

   export LD_LIBRARY_PATH=/lib64/lib:$LD_LIBRARY_PATH

3. Set LD_LIBRARY_PATH to include the Advanced Server Open Client library. By default, libiconv.so.2 is located in $EDB_HOME/oci/lib.

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

4. Then, compile and link the OCL API program.

   cd $EDB_HOME/oci/samples
   
   make
3.4 Ref Cursor Support

The Advanced Server Open Client Library supports the use of REF CURSOR as OUT parameters in PL/SQL procedures that are compatible with Oracle. Support is provided through the following APIs:

- OCIBindByName
- OCIBindByPos
- OCIBindDynamic
- OCISqlStmtPrepare
- OCISqlStmtExecute
- OCISqlStmtFetch
- OCIAtrrGet

The EDB OCL connector also supports the SQLT_RST data type.

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:

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

Allocate handles for executing a stored procedure to open and use the REF CURSOR:

```
Then, prepare the PL/SQL block that is used to open the REF CURSOR:

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

Bind the PL/SQL openCursor OUT parameter:

```sql
OCIBindByPos(stmtOpenRefCursor,
    &bndplrc1,
    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));
```

Use the stmtOpenRefCursor statement handle to call the openCursor procedure:

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

At this point, the stmtUseRefCursor statement handle contains the reference to the cursor. To obtain the information, define output variables for the ref cursor:

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

(continues on next page)
(continued from previous page)

```c
(sb4) sizeof(empNo),
SQLT_INT,
(void *) 0,
(ub2 *) 0,
(ub2 *) 0,
(ub4) OCI_DEFAULT));
```

Then, fetch the first row of the result set into the target variables:

```c
/* Fetch the cursor data */
OCISstmtFetch(stmtUseRefCursor,
errhp,
(ub4) 1,
(ub4) OCI_FETCH_NEXT,
(ub4) OCI_DEFAULT))
```
3.5 OCL Function Reference

The following tables list the functions supported by the EDB OCL connector. Note that any and all header files must be supplied by the user. Advanced Server does not supply any such files.

3.5.1 Connect, Authorize and Initialize Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIBreak</td>
<td>Aborts the specified OCL function.</td>
</tr>
<tr>
<td>OCIEnvCreate</td>
<td>Creates an OCL environment.</td>
</tr>
<tr>
<td>OCIEnvInit</td>
<td>Initializes an OCL environment handle.</td>
</tr>
<tr>
<td>OCIInitialize</td>
<td>Initializes the OCL environment.</td>
</tr>
<tr>
<td>OCILogoff</td>
<td>Releases a session.</td>
</tr>
<tr>
<td>OCILogon</td>
<td>Creates a logon connection.</td>
</tr>
<tr>
<td>OCILogon2</td>
<td>Creates a logon session in various modes.</td>
</tr>
<tr>
<td>OCIReset</td>
<td>Resets the current operation/protocol.</td>
</tr>
<tr>
<td>OCIServerAttach</td>
<td>Establishes an access path to a data source.</td>
</tr>
<tr>
<td>OCIServerDetach</td>
<td>Removes access to a data source.</td>
</tr>
<tr>
<td>OCISessionBegin</td>
<td>Creates a user session.</td>
</tr>
<tr>
<td>OCISessionEnd</td>
<td>Ends a user session.</td>
</tr>
<tr>
<td>OCISessionGet</td>
<td>Gets session from session pool.</td>
</tr>
<tr>
<td>OCISessionRelease</td>
<td>Releases a session.</td>
</tr>
<tr>
<td>OCITerminate</td>
<td>Detaches from shared memory subsystem.</td>
</tr>
</tbody>
</table>

Using the tnsnames.ora File

The OCIServerAttach and OCILogon method uses NET_SERVICE_NAME as connection descriptor specified in the dblink parameter of the tnsnames.ora file. Use the tnsnames.ora file (compatible with Oracle databases), to specify database connection details. OCL searches the user’s home directory for a file named .tnsnames.ora; if OCL doesn’t find the .tnsnames.ora file in the user’s home directory, it searches the tnsnames.ora on path specified in TNS_ADMIN environment variable.

Multiple descriptors (NET_SERVICE_NAME) can be specified in tnsnames.ora file.

The sample tnsnames.ora file contains:

```
EDBX =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 5444))
   (CONNECT_DATA = (SERVER = DEDICATED)(SID = edb))
  )
```
Any parameters not included in the files are ignored by the Open Client Library. In the example, SID refers to the database named edb, in the cluster running on the localhost on port 5444.

A C program call to OCIServerAttach that uses the tnsnames.ora file will look like:

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

If you don’t have a tnsnames.ora file, supply the connection string in the form //localhost:5444/edb.

**Note:** Multiple Descriptors are also supported in tnsnames.ora.
## 3.5.2 Handle and Descriptor Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIAttrGet</td>
<td>Get handle attributes. Advanced server supports the following handle attributes: OCI_ATTR_USERNAME, OCI_ATTR_PASSWORD, OCI_ATTR_SERVER, OCI_ATTR_ENV, OCI_ATTR_SESSION, OCI_ATTR_ROW_COUNT, OCI_ATTR_CHARSET_FORM, EDB_ATTR_STMT_LEVEL_TX, OCI_ATTR_MODULE</td>
</tr>
<tr>
<td>OCIAttrSet</td>
<td>Set handle attributes. Advanced server supports the following handle attributes: 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_MODULE, OCI_ATTR_PREFETCH_ROWS</td>
</tr>
<tr>
<td>OCIDescriptorAlloc</td>
<td>Allocate and initialize a descriptor.</td>
</tr>
<tr>
<td>OCIDescriptorFree</td>
<td>Free an allocated descriptor.</td>
</tr>
<tr>
<td>OCIHandleAlloc</td>
<td>Allocate and initialize a handle.</td>
</tr>
<tr>
<td>OCIHandleFree</td>
<td>Free an allocated handle.</td>
</tr>
<tr>
<td>OCIParamGet</td>
<td>Get a parameter descriptor.</td>
</tr>
<tr>
<td>OCIParamSet</td>
<td>Set a parameter descriptor.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCI_DEFAULT</td>
<td>Treat an empty string as a NULL value.</td>
</tr>
<tr>
<td>EDB_EMPTY_STRINGS_NULL</td>
<td>Treat an empty string as a NULL value.</td>
</tr>
<tr>
<td>EDB_EMPTY_STRINGS_EMPTY</td>
<td>Treat an empty string as a string of zero length.</td>
</tr>
</tbody>
</table>

To find the value of EDB_ATTR_EMPTY_STRINGS, query OCIAttrGet().

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 - execute as a WITH HOLD cursor
- EDB_WITHOUT_HOLD - execute using a protocol-level prepared statement
- OCI_DEFAULT - see the definition that follows

You can set the attribute in an OCIStmt handle or an OCIServer handle. When you create an OCIServer handle or an OCIStmt handle, the EDB_ATTR_HOLDABLE attribute for that handle is set to OCI_DEFAULT.

You can change the EDB_ATTR_HOLDABLE attribute for a handle by calling OCIAttrSet() and retrieve the attribute by calling OCIAttrGet().

When Advanced Server executes a SELECT statement, it examines the EDB_ATTR_HOLDABLE attribute in the OCIServer handle. If that attribute is set to EDB_WITH_HOLD, the query is executed as a WITH HOLD cursor.

If the EDB_ATTR_HOLDABLE attribute in the OCIServer handle is set to EDB_WITHOUT_HOLD, the query is executed as a normal prepared statement.

If the EDB_ATTR_HOLDABLE attribute in the OCIServer handle is set to OCI_DEFAULT, Advanced Server uses the value of the EDB_ATTR_HOLDABLE attribute in the OCIServer handle (if the EDB_ATTR_HOLDABLE attribute in the OCIServer is set to EDB_WITH_HOLD, the query executes as a WITH HOLD cursor, otherwise, the query executes as a protocol-prepared statement).
EDB_HOLD_CURSOR_ACTION

The EDB_HOLD_CURSOR_ACTION attribute alters the way WITH HOLD cursors are created using the OCL interface. You can set this attribute to any of the following values:

- **EDB_COMMIT_AFTER_CURSOR** – commit the transaction after creating the cursor
- **EDB_CURSOR_WITHOUT_XACT_BLK** – do not begin a new transaction chain
- **OCI_DEFAULT** - see the definition that follows

The following describes the attribute values.

**OCI_DEFAULT**

Each time you execute a statement, the OCL examines the transaction state on the database server. If a transaction is not already in progress, the OCL executes a BEGIN statement to create a new transaction block, and then executes the statement that you provide. The transaction block remains open until you call OCITransCommit() or OCITransRollback().

By default, the database server closes any open cursors when you commit or rollback. If you (or the OCL) declare a cursor that includes the WITH HOLD clause, the cursor result set is persisted on the database server, and you may continue to fetch from that cursor. However, the database server will not persist open cursors when you roll back a transaction. If you try to fetch from a cursor after a ROLLBACK, the database server will report an error.

**EDB_COMMIT_AFTER_CURSOR**

If your application must read from a WITH HOLD cursor after rolling back a transaction, you can arrange for the OCL to commit the transaction immediately after creating the cursor by setting EDB_HOLD_CURSOR_ACTION to EDB_COMMIT_AFTER_CURSOR prior to creating such a cursor. For example:

```c
ub4 action = EDB_COMMIT_AFTER_CURSOR;

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

OCIStmtExecute( ... );
```

It is important to understand that using EDB_COMMIT_AFTER_CURSOR will commit any pending changes.

**EDB_CURSOR_WITHOUT_XACT_BLK**

If your application will not run properly with the extra commits added by EDB_COMMIT_AFTER_CURSOR, you may try setting EDB_ATTR_HOLD_CURSOR_ACTION to EDB_CURSOR_WITHOUT_XACT_BLK. With this action, the OCL will not begin a new transaction chain. If you create a WITH HOLD cursor immediately after committing or rolling back a transaction, the cursor will be created in its own transaction, the database server will commit that transaction, and the cursor will persist.
It is important to understand that you may still experience errors if the cursor declaration is not
the first statement within a transaction – if you execute some other statement before declaring the
cursor, the WITH HOLD cursor will be created in a transaction block and may be rolled back if an
error occurs (or if your application calls OCITransRollback()).

Please note that you can set the EDB_HOLD_CURSOR_ACTION on the server level (OCIServer)
or for each statement handle (OCIStmt). 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 at-
ttribute 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**

Unless otherwise instructed, the OCL connector will ROLLBACK the current transaction whenever
the server reports an error. If you choose, you can override the automatic ROLLBACK with the
edb_stmt_level_tx parameter, which preserves modifications within a transaction, even if
one (or several) statements raise an error within the transaction.

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():

```c
OCIServer *server = myServer;
ub1 enabled = 1;
OCIAttrSet(server, OCI_HTYPE_SERVER, &enabled,
    sizeof(enabled), EDB_ATTR_STMT_LEVEL_TX, err);
```

To disable edb_stmt_level_tx:

```c
OCIServer *server = myServer;
ub1 enabled = 0;
OCIAttrSet(server, OCI_HTYPE_SERVER, &enabled,
    sizeof(enabled), EDB_ATTR_STMT_LEVEL_TX, err);
```
3.5.3 Bind, Define and Describe Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIBindByName</td>
<td>Bind by name.</td>
</tr>
<tr>
<td>OCIBindByPos</td>
<td>Bind by position.</td>
</tr>
<tr>
<td>OCIBindDynamic</td>
<td>Set additional attributes after bind.</td>
</tr>
<tr>
<td>OCIBindArrayOfStruct</td>
<td>Bind an array of structures for bulk operations.</td>
</tr>
<tr>
<td>OCIDefineArrayOfStruct</td>
<td>Specify the attributes of an array.</td>
</tr>
<tr>
<td>OCIDefineByPos</td>
<td>Define an output variable association.</td>
</tr>
<tr>
<td>OCIDefineDynamic</td>
<td>Set additional attributes for define.</td>
</tr>
<tr>
<td>OCIDescribeAny</td>
<td>Describe existing schema objects.</td>
</tr>
<tr>
<td>OCIStmtGetBindInfo</td>
<td>Get bind and indicator variable names and handle.</td>
</tr>
<tr>
<td>OCIUserCallbackRegister</td>
<td>Define a user-defined callback.</td>
</tr>
</tbody>
</table>

3.5.4 Statement Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIStmtExecute</td>
<td>Execute a prepared SQL statement.</td>
</tr>
<tr>
<td>OCIStmtFetch</td>
<td>Fetch rows of data (deprecated).</td>
</tr>
<tr>
<td>OCIStmtFetch2</td>
<td>Fetch rows of data.</td>
</tr>
<tr>
<td>OCIStmtPrepare</td>
<td>Prepare a SQL statement.</td>
</tr>
<tr>
<td>OCIStmtPrepare2</td>
<td>Prepare a SQL statement.</td>
</tr>
<tr>
<td>OCIStmtRelease</td>
<td>Release a statement handle.</td>
</tr>
</tbody>
</table>

3.5.5 Transaction Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCITransCommit</td>
<td>Commit a transaction.</td>
</tr>
<tr>
<td>OCITransRollback</td>
<td>Roll back a transaction.</td>
</tr>
</tbody>
</table>
3.5.6 XA Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xaoEnv</td>
<td>Returns OCL environment handle.</td>
</tr>
<tr>
<td>xaoSvcCtx</td>
<td>Returns OCL service context.</td>
</tr>
</tbody>
</table>

**xaoSvcCtx**

In order to use the xaoSvcCtx function, extensions in the xaoSvcCtx or xa_open connection string format must be provided as follows:

Oracle_XA{+<required_fields> ...}

Where required_fields are the following:

- **HostName=host_ip_address** specifies the IP address of the Advanced Server database.
- **PortNumber=host_port_number** specifies the port number on which Advanced Server is running.
- **SqlNet=dbname** specifies the database name.
- **Acc=P/username/password** specifies the database username and password. *password* may be omitted in which case the field is specified as Acc=P/username/.
- **AppName=app_id** specifies a number that identifies the application.

The following is an example of the connection string:

Oracle_XA+HostName=192.168.1.1+PortNumber=1533+SqlNet=XE+Acc=P/user/ password+AppName=1234
### 3.5.7 Date and Datetime Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIDateAddDays</td>
<td>Add or subtract a number of days.</td>
</tr>
<tr>
<td>OCIDateAddMonths</td>
<td>Add or subtract a number of months.</td>
</tr>
<tr>
<td>OCIDateAssign</td>
<td>Assign a date.</td>
</tr>
<tr>
<td>OCIDateCheck</td>
<td>Check if the given date is valid.</td>
</tr>
<tr>
<td>OCIDateCompare</td>
<td>Compare two dates.</td>
</tr>
<tr>
<td>OCIDateDaysBetween</td>
<td>Find the number of days between two dates.</td>
</tr>
<tr>
<td>OCIDateFromText</td>
<td>Convert a string to a date.</td>
</tr>
<tr>
<td>OCIDateGetDate</td>
<td>Get the date portion of a date.</td>
</tr>
<tr>
<td>OCIDateGetTime</td>
<td>Get the time portion of a date.</td>
</tr>
<tr>
<td>OCIDateLastDay</td>
<td>Get the date of the last day of the month.</td>
</tr>
<tr>
<td>OCIDateNextDay</td>
<td>Get the date of the next day.</td>
</tr>
<tr>
<td>OCIDateSetDate</td>
<td>Set the date portion of a date.</td>
</tr>
<tr>
<td>OCIDateSetTime</td>
<td>Set the time portion of a date.</td>
</tr>
<tr>
<td>OCIDateSysDate</td>
<td>Get the current system date and time.</td>
</tr>
<tr>
<td>OCIDateToText</td>
<td>Convert a date to a string.</td>
</tr>
<tr>
<td>OCIDateTimeAssign</td>
<td>Perform datetime assignment.</td>
</tr>
<tr>
<td>OCIDateTimeCheck</td>
<td>Check if the date is valid.</td>
</tr>
<tr>
<td>OCIDateTimeCompare</td>
<td>Compare two datetime values.</td>
</tr>
<tr>
<td>OCIDateTimeConstruct</td>
<td>Construct a datetime descriptor.</td>
</tr>
<tr>
<td>OCIDateTimeConvert</td>
<td>Convert one datetime type to another.</td>
</tr>
<tr>
<td>OCIDateTimeFromArray</td>
<td>Convert an array of size OCI_DT_ARRAYLEN to an OCIDateTime de-</td>
</tr>
<tr>
<td></td>
<td>scriptor.</td>
</tr>
<tr>
<td>OCIDateTimeFromText</td>
<td>Convert the given string to Oracle datetime type in the OCIDate-</td>
</tr>
<tr>
<td></td>
<td>Time descriptor according to the specified format.</td>
</tr>
<tr>
<td>OCIDateTimeGetDate</td>
<td>Get the date portion of a datetime value.</td>
</tr>
<tr>
<td>OCIDateTimeGetTime</td>
<td>Get the time portion of a datetime value.</td>
</tr>
<tr>
<td>OCIDateTimeGetTimeZoneName</td>
<td>Get the time zone name portion of a datetime value.</td>
</tr>
<tr>
<td>OCIDateTimeGetTimeZoneOffset</td>
<td>Get the time zone (hour, minute) portion of a datetime value.</td>
</tr>
<tr>
<td>OCIDateTimeSubtract</td>
<td>Take two datetime values as input and return their difference</td>
</tr>
<tr>
<td></td>
<td>as an interval.</td>
</tr>
<tr>
<td>OCIDateTimeSysTimeStamp</td>
<td>Get the system current date and time as a timestamp with time</td>
</tr>
<tr>
<td></td>
<td>zone.</td>
</tr>
<tr>
<td>OCIDateTimeToArray</td>
<td>Convert an OCIDateTime descriptor to an array.</td>
</tr>
<tr>
<td>OCIDateTimeToText</td>
<td>Convert the given date to a string according to the specified</td>
</tr>
<tr>
<td></td>
<td>format.</td>
</tr>
</tbody>
</table>
### 3.5.8 Interval Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIIntervalAdd</td>
<td>Adds two interval values.</td>
</tr>
<tr>
<td>OCIIntervalAssign</td>
<td>Copies one interval value into another interval value.</td>
</tr>
<tr>
<td>OCIIntervalCompare</td>
<td>Compares two interval values.</td>
</tr>
<tr>
<td>OCIIntervalGetDaySecond</td>
<td>Extracts days, hours, minutes, seconds and fractional seconds from an interval.</td>
</tr>
<tr>
<td>OCIIntervalSetDaySecond</td>
<td>Modifies days, hours, minutes, seconds and fractional seconds in an interval.</td>
</tr>
<tr>
<td>OCIIntervalGetYearMonth</td>
<td>Extracts year and month values from an interval.</td>
</tr>
<tr>
<td>OCIIntervalSetYearMonth</td>
<td>Modifies year and month values in an interval.</td>
</tr>
<tr>
<td>OCIIntervalDivide</td>
<td>Implements division of OCIInterval values by OCINumber values.</td>
</tr>
<tr>
<td>OCIIntervalMultiply</td>
<td>Implements multiplication of OCIInterval values by OCINumber values.</td>
</tr>
<tr>
<td>OCIIntervalSubtract</td>
<td>Subtracts one interval value from another interval value.</td>
</tr>
<tr>
<td>OCIIntervalToText</td>
<td>Extrapolates a character string from an interval.</td>
</tr>
<tr>
<td>OCIIntervalCheck</td>
<td>Verifies the validity of an interval value.</td>
</tr>
<tr>
<td>OCIIntervalToNumber</td>
<td>Converts an OCIInterval value into a OCINumber value.</td>
</tr>
<tr>
<td>OCIIntervalFromNumber</td>
<td>Converts a OCINumber value into an OCIInterval value.</td>
</tr>
<tr>
<td>OCIDateTimeIntervalAdd</td>
<td>Adds an OCIInterval value to an OCIDatetime value, resulting in an OCIDatetime value.</td>
</tr>
<tr>
<td>OCIDateTimeIntervalSub</td>
<td>Subtracts an OCIInterval value from an OCIDatetime value, resulting in an OCIDatetime value.</td>
</tr>
<tr>
<td>OCIIntervalFromText</td>
<td>Converts a text string into an interval.</td>
</tr>
<tr>
<td>OCIIntervalFromTZ</td>
<td>Converts a time zone specification into an interval value.</td>
</tr>
</tbody>
</table>
### 3.5.9 Number Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCINumberAbs</td>
<td>Compute the absolute value.</td>
</tr>
<tr>
<td>OCINumberAdd</td>
<td>Adds NUMBERs.</td>
</tr>
<tr>
<td>OCINumberArcCos</td>
<td>Compute the arc cosine.</td>
</tr>
<tr>
<td>OCINumberArcSin</td>
<td>Compute the arc sine.</td>
</tr>
<tr>
<td>OCINumberArcTan</td>
<td>Compute the arc tangent.</td>
</tr>
<tr>
<td>OCINumberArcTan2</td>
<td>Compute the arc tangent of two NUMBERs.</td>
</tr>
<tr>
<td>OCINumberAssign</td>
<td>Assign one NUMBER to another.</td>
</tr>
<tr>
<td>OCINumberCeil</td>
<td>Compute the ceiling of NUMBER.</td>
</tr>
<tr>
<td>OCINumberCmp</td>
<td>Compare NUMBERs.</td>
</tr>
<tr>
<td>OCINumberCos</td>
<td>Compute the cosine.</td>
</tr>
<tr>
<td>OCINumberDec</td>
<td>Decrement a NUMBER.</td>
</tr>
<tr>
<td>OCINumberDiv</td>
<td>Divide two NUMBERs.</td>
</tr>
<tr>
<td>OCINumberExp</td>
<td>Raise e to the specified NUMBER power.</td>
</tr>
<tr>
<td>OCINumberFloor</td>
<td>Compute the floor of a NUMBER.</td>
</tr>
<tr>
<td>OCINumberFromInt</td>
<td>Convert an integer to an Oracle NUMBER.</td>
</tr>
<tr>
<td>OCINumberFromReal</td>
<td>Convert a real to an Oracle NUMBER.</td>
</tr>
<tr>
<td>OCINumberFromText</td>
<td>Convert a string to an Oracle NUMBER.</td>
</tr>
<tr>
<td>OCINumberHypCos</td>
<td>Compute the hyperbolic cosine.</td>
</tr>
<tr>
<td>OCINumberHypSin</td>
<td>Compute the hyperbolic sine.</td>
</tr>
<tr>
<td>OCINumberHypTan</td>
<td>Compute the hyperbolic tangent.</td>
</tr>
<tr>
<td>OCINumberInc</td>
<td>Increments a NUMBER.</td>
</tr>
<tr>
<td>OCINumberIntPower</td>
<td>Raise a given base to an integer power.</td>
</tr>
<tr>
<td>OCINumberIsInt</td>
<td>Test if a NUMBER is an integer.</td>
</tr>
<tr>
<td>OCINumberIsZero</td>
<td>Test if a NUMBER is zero.</td>
</tr>
<tr>
<td>OCINumberLn</td>
<td>Compute the natural logarithm.</td>
</tr>
<tr>
<td>OCINumberLog</td>
<td>Compute the logarithm to an arbitrary base.</td>
</tr>
<tr>
<td>OCINumberMod</td>
<td>Modulo division.</td>
</tr>
<tr>
<td>OCINumberMul</td>
<td>Multiply NUMBERs.</td>
</tr>
<tr>
<td>OCINumberNeg</td>
<td>Negate a NUMBER.</td>
</tr>
<tr>
<td>OCINumberPower</td>
<td>Exponentiation to base e.</td>
</tr>
<tr>
<td>OCINumberPrec</td>
<td>Round a NUMBER to a specified number of decimal places.</td>
</tr>
<tr>
<td>OCINumberRound</td>
<td>Round a NUMBER to a specified decimal place.</td>
</tr>
<tr>
<td>OCINumberSetPi</td>
<td>Initialize a NUMBER to Pi.</td>
</tr>
<tr>
<td>OCINumberSetZero</td>
<td>Initialize a NUMBER to zero.</td>
</tr>
<tr>
<td>OCINumberShift</td>
<td>Multiply by 10, shifting specified number of decimal places.</td>
</tr>
<tr>
<td>OCINumberSign</td>
<td>Obtain the sign of a NUMBER.</td>
</tr>
<tr>
<td>OCINumberSin</td>
<td>Compute the sine.</td>
</tr>
<tr>
<td>OCINumberSqrt</td>
<td>Compute the square root of a NUMBER.</td>
</tr>
<tr>
<td>OCINumberSub</td>
<td>Subtract NUMBERs.</td>
</tr>
</tbody>
</table>

*continues on next page*
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCINumberTan</td>
<td>Compute the tangent.</td>
</tr>
<tr>
<td>OCINumberToInt</td>
<td>Convert a NUMBER to an integer.</td>
</tr>
<tr>
<td>OCINumberToReal</td>
<td>Convert a NUMBER to a real.</td>
</tr>
<tr>
<td>OCINumberToRealArray</td>
<td>Convert an array of NUMBER to a real array.</td>
</tr>
<tr>
<td>OCINumberToText</td>
<td>Converts a NUMBER to a string.</td>
</tr>
<tr>
<td>OCINumberTrunc</td>
<td>Truncate a NUMBER at a specified decimal place.</td>
</tr>
</tbody>
</table>
### 3.5.10 String Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIStringAllocSize</td>
<td>Get allocated size of string memory in bytes.</td>
</tr>
<tr>
<td>OCIStringAssign</td>
<td>Assign string to a string.</td>
</tr>
<tr>
<td>OCIStringAssignText</td>
<td>Assign text string to a string.</td>
</tr>
<tr>
<td>OCIStringPtr</td>
<td>Get string pointer.</td>
</tr>
<tr>
<td>OCIStringResize</td>
<td>Resize string memory.</td>
</tr>
<tr>
<td>OCIStringSize</td>
<td>Get string size.</td>
</tr>
</tbody>
</table>

### 3.5.11 Cartridge Services and File I/O Interface Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIFileClose</td>
<td>Close an open file.</td>
</tr>
<tr>
<td>OCIFileExists</td>
<td>Test to see if the file exists.</td>
</tr>
<tr>
<td>OCIFileFlush</td>
<td>Write buffered data to a file.</td>
</tr>
<tr>
<td>OCIFileGetLength</td>
<td>Get the length of a file.</td>
</tr>
<tr>
<td>OCIFileInit</td>
<td>Initialize the OCIFile package.</td>
</tr>
<tr>
<td>OCIFileOpen</td>
<td>Open a file.</td>
</tr>
<tr>
<td>OCIFileRead</td>
<td>Read from a file into a buffer.</td>
</tr>
<tr>
<td>OCIFileSeek</td>
<td>Change the current position in a file.</td>
</tr>
<tr>
<td>OCIFileTerm</td>
<td>Terminate the OCIFile package.</td>
</tr>
<tr>
<td>OCIFileWrite</td>
<td>Write buflen bytes into the file.</td>
</tr>
</tbody>
</table>

### 3.5.12 LOB Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCILobRead</td>
<td>Returns a LOB value (or a portion of a LOB value).</td>
</tr>
<tr>
<td>OCILOBWriteAppend</td>
<td>Adds data to a LOB value.</td>
</tr>
<tr>
<td>OCILobGetLength</td>
<td>Returns the length of a LOB value.</td>
</tr>
<tr>
<td>OCILobTrim</td>
<td>Trims data from the end of a LOB value.</td>
</tr>
<tr>
<td>OCILobOpen</td>
<td>Opens a LOB value for use by other LOB functions.</td>
</tr>
<tr>
<td>OCILobClose</td>
<td>Closes a LOB value.</td>
</tr>
</tbody>
</table>
### 3.5.13 Miscellaneous Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIClientVersion</td>
<td>Return client library version.</td>
</tr>
<tr>
<td>OCIErrorGet</td>
<td>Return error message.</td>
</tr>
<tr>
<td>OCIPGErrorGet</td>
<td>Return native error messages reported by libpq or the server. The signature is: sword OCIPGErrorGet(dvoid *hndlp, ub4 recordno, OraText *errcodep, ub4 errbufsiz, OraText *bufp, ub4 bufsize, ub4 type)</td>
</tr>
<tr>
<td>OCIPasswordChange</td>
<td>Change password.</td>
</tr>
<tr>
<td>OCIPing</td>
<td>Confirm that the connection and server are active.</td>
</tr>
<tr>
<td>OCIServerVersion</td>
<td>Get the Oracle version string.</td>
</tr>
</tbody>
</table>
### 3.5.14 Supported Data Types

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI_DATE</td>
<td>ANSI date</td>
</tr>
<tr>
<td>SQLT_AFC</td>
<td>ANSI fixed character</td>
</tr>
<tr>
<td>SQLT_AVC</td>
<td>ANSI variable character</td>
</tr>
<tr>
<td>SQLT_BDOUBLE</td>
<td>Binary double</td>
</tr>
<tr>
<td>SQLT_BIN</td>
<td>Binary data</td>
</tr>
<tr>
<td>SQLT_BFLOAT</td>
<td>Binary float</td>
</tr>
<tr>
<td>SQLT_CHR</td>
<td>Character string</td>
</tr>
<tr>
<td>SQLT_DAT</td>
<td>Oracle date</td>
</tr>
<tr>
<td>SQLT_DATE</td>
<td>ANSI date</td>
</tr>
<tr>
<td>SQLT_FLT</td>
<td>Float</td>
</tr>
<tr>
<td>SQLT_INT</td>
<td>Integer</td>
</tr>
<tr>
<td>SQLT_LBI</td>
<td>Long binary</td>
</tr>
<tr>
<td>SQLT_LNG</td>
<td>Long</td>
</tr>
<tr>
<td>SQLT_LVB</td>
<td>Longer long binary</td>
</tr>
<tr>
<td>SQLT_LVC</td>
<td>Longer longs (character)</td>
</tr>
<tr>
<td>SQLT_NUM</td>
<td>Oracle numeric</td>
</tr>
<tr>
<td>SQLT_ODT</td>
<td>OCL date type</td>
</tr>
<tr>
<td>SQLT_STR</td>
<td>Zero-terminated string</td>
</tr>
<tr>
<td>SQLT_TIMESTAMP</td>
<td>Timestamp</td>
</tr>
<tr>
<td>SQLT_TIMESTAMP_TZ</td>
<td>Timestamp with time zone</td>
</tr>
<tr>
<td>SQLT_TIMESTAMP_LTZ</td>
<td>Timestamp with local time zone</td>
</tr>
<tr>
<td>SQLT_UIN</td>
<td>Unsigned integer</td>
</tr>
<tr>
<td>SQLT_VBI</td>
<td>VCS format binary</td>
</tr>
<tr>
<td>SQLT_VCS</td>
<td>Variable character</td>
</tr>
<tr>
<td>SQLT_VNU</td>
<td>Number with preceding length byte</td>
</tr>
<tr>
<td>SQLT_VST</td>
<td>OCL string type</td>
</tr>
</tbody>
</table>
## 3.6 OCL Error Codes – Reference

The following table lists the error code mappings defined by the OCL Connector. When the database server reports an error code or condition (shown in the first or second column), the OCL converts the value to the compatible value displayed in the third column.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Condition Name</th>
<th>Oracle Error Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>42601</td>
<td>syntax_error</td>
<td>ORA-16945</td>
</tr>
<tr>
<td>42P01</td>
<td>undefined_table</td>
<td>ORA-00942</td>
</tr>
<tr>
<td>02000</td>
<td>no_data</td>
<td>ORA-01403</td>
</tr>
<tr>
<td>08000</td>
<td>connection_exception</td>
<td>ORA-12545</td>
</tr>
<tr>
<td>08003</td>
<td>connection_does_not_exist</td>
<td>ORA-12545</td>
</tr>
<tr>
<td>08006</td>
<td>connection_failure</td>
<td>ORA-12545</td>
</tr>
<tr>
<td>08001</td>
<td>sqlclient_unable_to_establish_sqlconnection</td>
<td>ORA-12545</td>
</tr>
<tr>
<td>08004</td>
<td>sqlserver_rejected_establishment_of_sqlconnection</td>
<td>ORA-12545</td>
</tr>
<tr>
<td>25000</td>
<td>invalid_transaction_state</td>
<td>ORA-01453</td>
</tr>
<tr>
<td>08007</td>
<td>transaction_resolution_unknown</td>
<td>ORA-01453</td>
</tr>
<tr>
<td>0A000</td>
<td>feature_not_supported</td>
<td>ORA-03001</td>
</tr>
<tr>
<td>22012</td>
<td>division_by_zero</td>
<td>ORA-01476</td>
</tr>
<tr>
<td>2200B</td>
<td>escape_character_conflict</td>
<td>ORA-01424</td>
</tr>
<tr>
<td>22019</td>
<td>invalid_escape_character</td>
<td>ORA-00911</td>
</tr>
<tr>
<td>2200D</td>
<td>invalid_escape_octet</td>
<td>ORA-01424</td>
</tr>
<tr>
<td>22025</td>
<td>invalid_escape_sequence</td>
<td>ORA-01424</td>
</tr>
<tr>
<td>22P06</td>
<td>nonstandard_use_of_escape_character</td>
<td>ORA-01424</td>
</tr>
<tr>
<td>2200C</td>
<td>invalid_use_of_escape_character</td>
<td>ORA-01424</td>
</tr>
<tr>
<td>22004</td>
<td>null_value_not_allowed</td>
<td>ORA-01400</td>
</tr>
<tr>
<td>23000</td>
<td>integrity_constraint_violation</td>
<td>ORA-00001</td>
</tr>
<tr>
<td>23505</td>
<td>unique_violation</td>
<td>ORA-00001</td>
</tr>
<tr>
<td>40P01</td>
<td>t_r_deadlock_detected</td>
<td>ORA-00060</td>
</tr>
<tr>
<td>42701</td>
<td>duplicate_column</td>
<td>ORA-01430</td>
</tr>
<tr>
<td>53000</td>
<td>insufficient_resources</td>
<td>ORA-01659</td>
</tr>
<tr>
<td>53100</td>
<td>disk_full</td>
<td>ORA-01659</td>
</tr>
<tr>
<td>53200</td>
<td>out_of_memory</td>
<td>ORA-01659</td>
</tr>
<tr>
<td>42P07</td>
<td>duplicate_table</td>
<td>ORA-00955</td>
</tr>
<tr>
<td>21000</td>
<td>cardinality_violation</td>
<td>ORA-01427</td>
</tr>
<tr>
<td>22003</td>
<td>numeric_value_out_of_range</td>
<td>ORA-01426</td>
</tr>
<tr>
<td>22P02</td>
<td>invalid_text_representation</td>
<td>ORA-01858</td>
</tr>
<tr>
<td>28000</td>
<td>invalid_authorization_specification</td>
<td>ORA-01017</td>
</tr>
<tr>
<td>28P01</td>
<td>invalid_password</td>
<td>ORA-01017</td>
</tr>
<tr>
<td>2200F</td>
<td>zero_length_character_string</td>
<td>ORA-01425</td>
</tr>
<tr>
<td>42704</td>
<td>undefined_object</td>
<td>ORA-01418</td>
</tr>
<tr>
<td>2BP01</td>
<td>dependent_objects_still_exist</td>
<td>ORA-02429</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Condition Name</th>
<th>Oracle Error Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>22027</td>
<td>trim_error</td>
<td>ORA-30001</td>
</tr>
<tr>
<td>22001</td>
<td>string_data_right_truncation</td>
<td>ORA-01401</td>
</tr>
<tr>
<td>22002</td>
<td>null_value_no_indicator_parameter</td>
<td>ORA-01405</td>
</tr>
<tr>
<td>22008</td>
<td>datetime_field_overflow</td>
<td>ORA-01800</td>
</tr>
<tr>
<td>44000</td>
<td>with_check_optionViolation</td>
<td>ORA-01402</td>
</tr>
<tr>
<td>01007</td>
<td>warning_privilege_not_granted</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>01006</td>
<td>warning_privilege_not_revoked</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>02001</td>
<td>no_additional_dynamic_result_sets_returned</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>03000</td>
<td>sql_statement_not_yet_complete</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>08P01</td>
<td>protocol_violation</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>23001</td>
<td>restrict_violation</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>23502</td>
<td>not_null_violation</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>23505</td>
<td>foreign_key_violation</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>23514</td>
<td>check_violation</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>24000</td>
<td>invalid_cursor_state</td>
<td>ORA-01001</td>
</tr>
<tr>
<td>26000</td>
<td>invalid_sql_statement_name</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>42830</td>
<td>invalid_foreign_key</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>55006</td>
<td>object_in_use</td>
<td>ORA-00000</td>
</tr>
<tr>
<td>55P03</td>
<td>lock_not_available</td>
<td>ORA-00054</td>
</tr>
<tr>
<td>72000</td>
<td>snapshot_too_old</td>
<td>ORA-01555</td>
</tr>
</tbody>
</table>

For more information about Postgres error codes, please see the PostgreSQL core documentation.
3.7 Multithreading Support

OCL is supported in multithreaded environment. You can enable/use multithreading in a multithreaded environment by making an `OCIEnvNlsCreate()` call with `OCI_THREADED` as the value of the mode parameter.

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

All subsequent calls to `OCIEnvNlsCreate()` must also be made with `OCI_THREADED`.

OCI library manages mutexes for the application for each environment handle if a multithreaded application is running on a thread-safe operating system.
CHAPTER 4

Generating the OCL Trace

The OCL tracing option logs direct communication (queries, updates, etc.) with the backend in specified OCI_DEBUG_LOG file. In addition, it also logs the functions/APIs that were invoked. The trace files are generated in the default working directory (oci_log_file_name). If you append the path with a file name (directory path/oci_log_file_name), then the trace files are generated at specific location.

A tracefile is generated for each connection in text file (readable) format.

**Note:** OCL tracing is disabled by default.

To generate the OCL Trace:

1. Enable the EDB Client Side tracing for OCL. You can enable the OCL tracing by setting below environment variables:

   ```bash
   export OCI_DEBUG_LEVEL=4
   export OCI_DEBUG_LOG=oci_log_file
   ```

2. Once you have exported the environment variables, execute the application. The OCL trace files are generated in the specified directory.
CHAPTER 5

Security and Encryption

5.1 Using SSL

EDB Postgres Advanced Server provides native support for using SSL connections to encrypt client/server communications for increased security. In OCL, it is controlled by setting the sslmode parameter to verify-full or verify-ca, and providing the system with a root certificate to verify against.

Steps of SSL configuration:

1. Configure the Server and Client Side Certificates; for detailed information about configuring SSL client and server side certificates, refer to the PostgreSQL SSL documentation.

2. Enable the SSL OCL Connection:

   In an OCL client application, you can enable SSL mode by setting the EDB_ATTR_SSL attribute in Session.

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

   Note: EDB_ATTR_SSL is defined in edboci.h header file available in installation directory.

3. After setting SSL attribute, you can use the OCIILogon function to create a connection:
Once the server is authenticated, then the client is ready to pass sensitive data.

For more information about the supported SSL mode options, please see:

https://www.postgresql.org/docs/13/libpq-ssl.html#LIBPQ-SSL-SSLMODE-STATEMENTS

### 5.2 Scram Compatibility

The EDB OCL driver provides SCRAM-SHA-256 support for Advanced Server version 11 and onwards. This support is available from EDB OCL 11.0.1 release onwards.
CHAPTER 6

Conclusion

EDB OCL Connector Guide
Copyright © 2007 - 2020 EnterpriseDB Corporation.
All rights reserved.
EnterpriseDB® Corporation
34 Crosby Drive, Suite 201, Bedford, MA 01730, USA
T +1 781 357 3390 F +1 978 467 1307 E
info@enterprisedb.com
www.enterprisedb.com

- EnterpriseDB and Postgres Enterprise Manager are registered trademarks of EnterpriseDB Corporation. EDB and EDB Postgres are trademarks of EnterpriseDB Corporation. Oracle is a registered trademark of Oracle, Inc. Other trademarks may be trademarks of their respective owners.

- EDB designs, establishes coding best practices, reviews, and verifies input validation for the logon UI for EDB products where present. EDB follows the same approach for additional input components, however the nature of the product may require that it accepts freeform SQL, WMI or other strings to be entered and submitted by trusted users for which limited validation is possible. In such cases it is not possible to prevent users from entering incorrect or otherwise dangerous inputs.

- EDB reserves the right to add features to products that accept freeform SQL, WMI or other potentially dangerous inputs from authenticated, trusted users in the future, but will ensure
all such features are designed and tested to ensure they provide the minimum possible risk, and where possible, require superuser or equivalent privileges.

- EDB does not warrant that we can or will anticipate all potential threats and therefore our process cannot fully guarantee that all potential vulnerabilities have been addressed or considered.
Index

A
Authorize Functions, 28

B
Bind Functions, 34

C
Cartridge Services and File I/O Interface Functions, 40
centos7, 10
centos8, 12
Compiling and Linking a Program, 24
compiling and linking program, 24
Conclusion, 49
Connect Functions, 28

D
Date and Datetime Functions, 36
Define Functions, 34
Describe Functions, 34

E
EDB_ATTR_HOLDABLE, 31
EDB_ATTR_STMT_LVL_TX, 33
EDB_HOLD_CURSOR_ACTION, 31

F
Forming a Connection String, 23

G
Generating the OCL Trace, 46
graphical installation, 17

H
Handle and Descriptor Functions, 30

I
Initialize Functions, 28
Installation on Debian, 15
Installation on SLES 12, 14
Installation on Ubuntu, 15
Installing and Configuring the OCL Connector, 6
Interval Functions, 37

L
LOB Functions, 40

M
Miscellaneous Functions, 41
Multithreading Support, 45

N
Number Functions, 38

O
OCL Error Codes – Reference, 43
OCL Function Reference, 28
Open Client Library, 5
Ref Cursor Support, 25
Ref cursor support, 25
rhel8, 8

51
S
Scram Compatibility, 48
Security and Encryption, 47
Statement Functions, 34
String Functions, 40
Supported Data Types, 42
Supported Platforms, 3
Supported server versions, 3

T
tnsnames.ora, 28
Transaction Functions, 34

U
Using SSL, 47

W
What’s New, 2

X
XA Functions, 35
xaoSvcCtx, 35