EDB Postgres™ Advanced Server OCI Connector Guide

Connectors Release 10.0.2
OCI Connector Version 10.0.2.0

April 24, 2018

EDB Postgres™ Advanced Server OCI Connector Guide
by EnterpriseDB® Corporation

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

Table of Contents

1 Introduction

The OCI connector provides an API similar to the Oracle Call Interface. Applications that are written to use the Oracle Call Interface may be recompiled using EnterpriseDB's OCI connector in order to interact with an Advanced Server database server.

This guide provides installation and usage instructions:

- 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.

Please note: EnterpriseDB 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 EnterpriseDB product.

1.1 Typographical Conventions Used in this Guide

Certain typographical conventions are used in this manual to clarify the meaning and usage of various commands, statements, programs, examples, etc. This section provides a summary of these conventions.

In the following descriptions a *term* refers to any word or group of words which may be language keywords, user-supplied values, literals, etc. A term's exact meaning depends upon the context in which it is used.

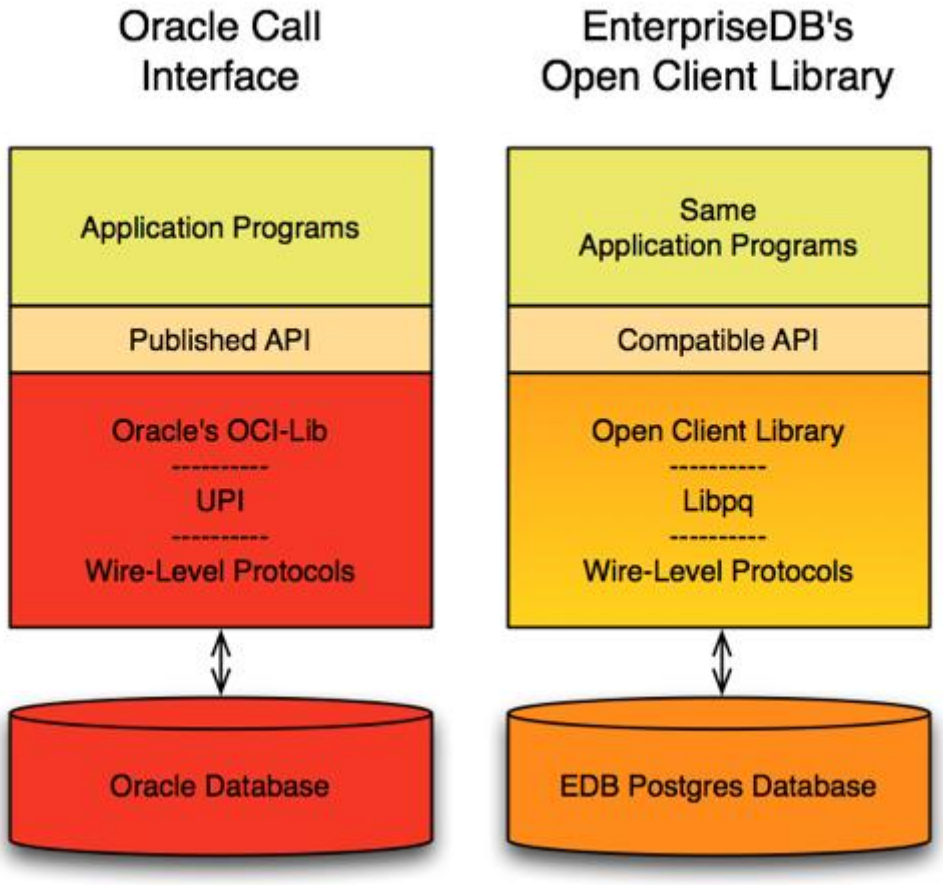
- *Italic font* introduces a new term, typically, in the sentence that defines it for the first time.
- `Fixed-width (mono-spaced) font` is used for terms that must be given literally such as SQL commands, specific table and column names used in the examples, programming language keywords, etc. For example, `SELECT * FROM emp;`
- *`Italic fixed-width font`* is used for terms for which the user must substitute values in actual usage. For example, `DELETE FROM` *`table_name`*;
- A vertical pipe | denotes a choice between the terms on either side of the pipe. A vertical pipe is used to separate two or more alternative terms within square brackets (optional choices) or braces (one mandatory choice).
- Square brackets [ ] denote that one or none of the enclosed term(s) may be substituted. For example, `[ a | b ]`, means choose one of "`a`" or "`b`" or neither of the two.
- Braces {} denote that exactly one of the enclosed alternatives must be specified. For example, `{ a | b }`, means exactly one of "`a`" or "`b`" must be specified.
- Ellipses ... denote that the proceeding term may be repeated. For example, `[ a | b ] ...` means that you may have the sequence, "`b a a b a`".

2 Open Client Library

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 EnterpriseDB implementation of the Open Client Library is written in C.

2.1 Comparison with Oracle Call Interface

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



2.2 Installing and Configuring the OCI Connector

You can use an RPM package or a graphical installer to install or update the OCI connector.

2.2.1 Installing the Connector with an RPM Package

Before using an RPM installer to install the OCI connector, you must first create and configure the EnterpriseDB repository file, providing a user name and password that allows access to the files in the repository. Contact a representative at EnterpriseDB for your download credentials.

The name of the package that creates the repository file for Advanced Server is `edb-repo`. You can download the package at:

http://yum.enterprisedb.com/

After downloading the repository configuration file, assume superuser privileges, and use the following command to create the repository configuration file:

```
rpm -Uvh edb-repo-x.x-x.noarch.rpm
```

The installer creates a repository configuration file named `edb.repo`; the file resides in `/etc/yum.repos.d`. The file contains entries for each of the EnterpriseDB repositories; to install the connector, you must enable the entry for the `enterprisedb-tools` repository:

```
[enterprisedb-tools]
name=EnterpriseDB Tools $releasever - $basearch
baseurl=http://<username>:<password>@yum.enterprisedb.com/tools/r
edhat/rhel-$releasever-$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Use your choice of editor to modify the repository configuration file:

1. Replace the *`user_name`* and *`password`* placeholders in the `baseurl` specification with your user name and the repository password.

2. Replace the `0` next to `enabled` with a `1`.

Save the configuration file and exit the editor.

After modifying the content of the repository configuration file, you can use the `yum install` command to install the connector; use the command:

```
yum install edb-oci
```

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.

2.2.2 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 EnterpriseDB.

Before installing the connector, use the following commands to add EnterpriseDB repository configuration files to your SLES host:

```
zypper addrepo https://zypp.enterprisedb.com/suse/epas96-
sles.repo
zypper addrepo https://zypp.enterprisedb.com/suse/epas-sles-
tools.repo
zypper addrepo https://zypp.enterprisedb.com/suse/epas-sles-
dependencies.repo
```

Each command creates a repository configuration file in the `/etc/zypp/repos.d` directory. The files are named:

- `edbas96suse.repo`
- `edbasdependencies.repo`
- `edbastools.repo`

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

```
/etc/zypp/repos.d # zypper refresh
Repository 'SLES12-12-0' is up to date.
Repository 'SLES12-Pool' is up to date.
Repository 'SLES12-Updates' is up to date.
Retrieving repository 'EDB Postgres Advanced Server 9.6 12 -
x86_64' metadata ----------------------[\]

Authentication required for
'https://zypp.enterprisedb.com/9.6/suse/suse-12-x86_64'
User Name:
Password:

Retrieving repository 'EDB Postgres Advanced Server 9.6 12 -
x86_64' metadata.................................[done]
Building repository 'EDB Postgres Advanced Server 9.6 12 -
x86_64' cache.........................[done]
All repositories have been refreshed.
...
```

When prompted for a `User Name` and `Password`, provide your connection credentials for the EnterpriseDB repository. If you need credentials, contact EnterpriseDB at:

https://www.enterprisedb.com/general-inquiry-form

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

For detailed information about registering a SUSE host, visit:

https://www.suse.com/support/kb/doc/?id=7016626

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

```
zypper install edb-oci

zypper install edb-oci-devel
```

2.2.3 Using the Graphical Installer to Install the Connector

You can use the EnterpriseDB Connectors Installation wizard to add the OCI connector to your system; the wizard is available at:

www.enterprisedb.com

This section demonstrates using the Installation Wizard to install the Connectors on a Linux system. (You can follow the same procedure 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.) The installation wizard opens as shown in Figure 2.1.

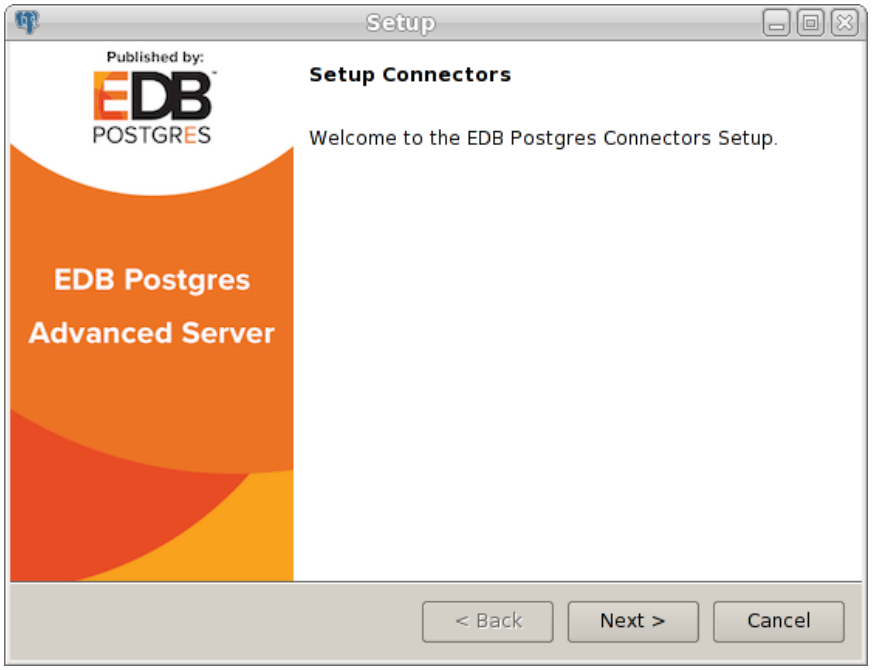


Figure 2.1 - The Connectors Installation wizard.

Click `Next` to continue.

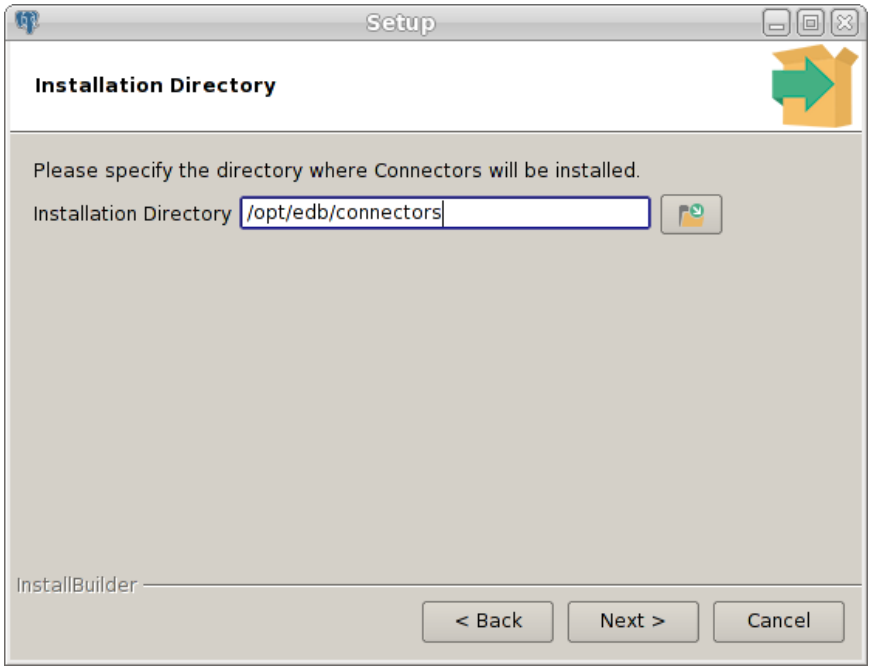


Figure 2.2 - The Installation dialog.

Use the `Installation Directory` dialog (see Figure 2.2) to specify the directory in which the connector will be installed, and click `Next` to continue.

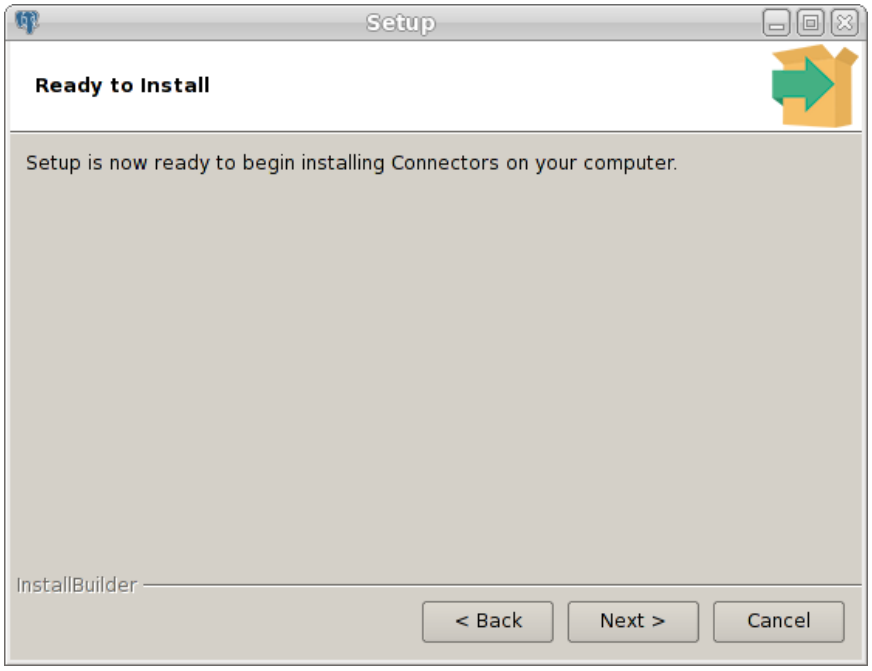


Figure 2.3 - The Ready to Install dialog.

Click `Next` on the `Ready to Install` dialog (see Figure 2.3) to start the installation; popup dialogs confirm the progress of the installation wizard.

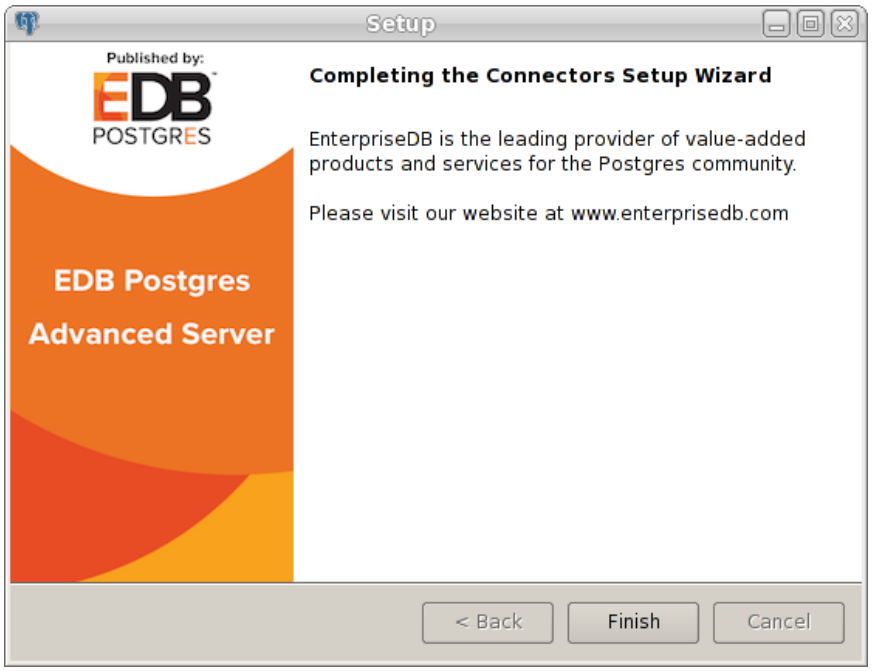


Figure 2.4 - The installation is complete.

When the wizard informs you that it has completed the setup, click the `Finish` button to exit the dialog (see Figure 2.4).

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 or through Linux `Applications` menu (see Figure 2.5).

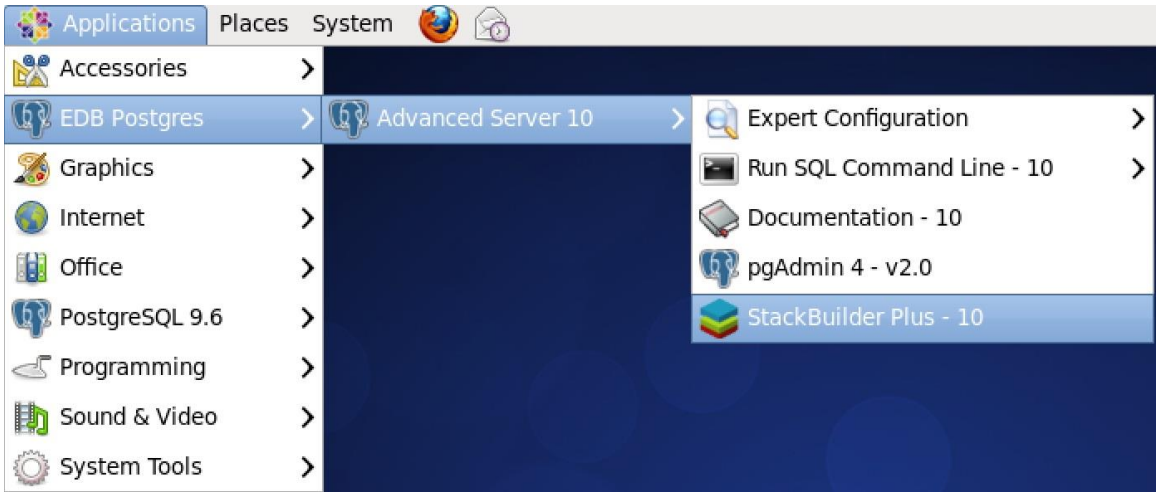


Figure 2.5 - Starting StackBuilder Plus.

When StackBuilder Plus opens, follow the onscreen instructions, selecting the `EnterpriseDB Connectors` option from the `Database Drivers` node of the tree control (see Figure 2.6).

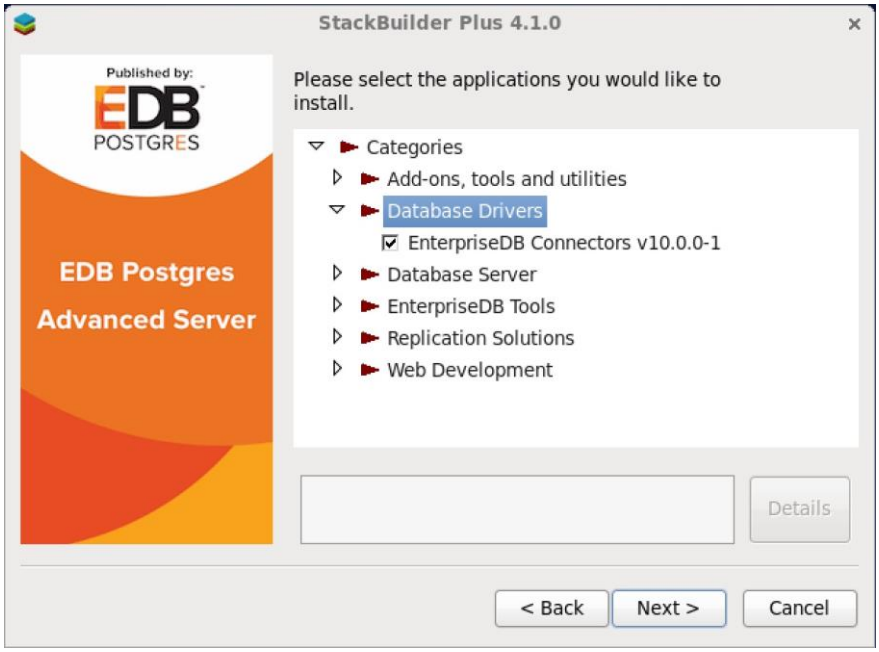


Figure 2.6 - Selecting the Connectors installer.

Follow the directions of the onscreen wizard to add or update an installation of the EnterpriseDB Connectors.

2.3 Forming a Connection String

The OCI 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://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://10.0.0.4:5444/edb
postgresql://10.0.0.4:5444/edb
```

For more information about using Postgres-style connection strings, please see the PostgreSQL core documentation, available at:

https://www.enterprisedb.com/resources/product-documentation

2.4 Compiling and Linking a Program

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

On Linux:

```
libedboci.so
```

On Windows:

```
edboci.dll
```

The files are installed in the `connectors/edb-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 `connectors/edb-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=/opt/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/connectors/edb-oci/lib`.

```
export
LD_LIBRARY_PATH=$EDB_HOME/connectors/edb-oci:$EDB_HOME/
connectors/edb-oci/lib:$LD_LIBRARY_PATH
```

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

```
cd $EDB_HOME/connectors/edb-oci/samples

make
```

2.5 Ref Cursor Support

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

- OCIBindByName
- OCIBindByPos
- OCIBindDynamic
- OCIStmtPrepare
- OCIStmtExecute
- OCIStmtFetch
- OCIAttrGet

The OCI connector also supports the SQLT_RSET 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:

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

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

```
  /* 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));
```

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

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

Bind the PL/SQL `openCursor OUT` parameter:

```
  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:

```
  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:

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



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

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

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


2.6 OCI Function Reference

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

2.6.1 Connect, Authorize and Initialize Functions

Table 9-2-1 Connect, Authorize, Terminate and Initialize Functions

Function	Description
OCIBreak	Aborts the specified OCI function.
OCIEnvCreate	Create an OCI environment.
OCIEnvInit	Initialize an OCI environment handle.
OCIInitialize	Initialize the OCI environment.
OCILogoff	Release a session.
OCILogon	Create a logon connection.
OCILogon2	Create a logon session in various modes.
OCIReset	Resets the current operation/protocol.
OCIServerAttach	Establish an access path to a data source.
OCIServerDetach	Remove access to a data source.
OCISessionBegin	Create a user session.
OCISessionEnd	End a user session.
OCISessionGet	Get session from session pool.
OCISessionRelease	Release a session.
OCITerminate	Detach from shared memory subsystem.

2.6.1.1 Using the tnsnames.ora File

The `OCIServerAttach` method uses a 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 addresses. Advanced Server searches the user's home directory for a file named `tnsnames.ora`. If Advanced Server doesn't find the `tnsnames.ora` file in the user's home directory, it searches the path specified by `TNS_ADMIN`.

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 sample, are ignored by the Open Client Library. In the sample, `SID` refers to the database named edb, in the cluster running on server 'localhost' at port `5444`.

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

```
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 parameter in the form //localhost:5444/edbx.

2.6.2 Handle and Descriptor Functions

Table 9-1 Handle and Descriptor Functions

Function	Description
OCIAttrGet	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, OCI_ATTR_CHARSET_ID, EDB_ATTR_STMT_LEVEL_TX, OCI_ATTR_MODULE
OCIAttrSet	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
OCIDescriptorAlloc	Allocate and initialize a descriptor.
OCIDescriptorFree	Free an allocated descriptor.
OCIHandleAlloc	Allocate and initialize a handle.
OCIHandleFree	Free an allocated handle.
OCIParamGet	Get a parameter descriptor.
OCIParamSet	Set a parameter descriptor.

2.6.2.1 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 OCI 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:

Value	Description
OCI_DEFAULT	Treat an empty string as a NULL value.
EDB_EMPTY_STRINGS_NULL	Treat an empty string as a NULL value.
EDB_EMPTY_STRINGS_EMPTY	Treat an empty string as a string of zero length.

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

2.6.2.2 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).

2.6.2.3 EDB_HOLD_CURSOR_ACTION

The EDB_HOLD_CURSOR_ACTION attribute alters the way WITH HOLD cursors are created using the OCI 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 OCI examines the transaction state on the database server. If a transaction is not already in progress, the OCI 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 OCI) 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 OCI 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:

```
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 OCI 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 attribute is set to `OCI_DEFAULT`), the value is taken from the server handle. So you can define a server-wide default action by setting the attribute in the server handle, and leaving the attribute set to `OCI_DEFAULT` in the statement handles. You can use different values for each statement handle (or server handle) as you see fit.

2.6.2.4 EDB_ATTR_STMT_LVL_TX

Unless otherwise instructed, the OCI 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()`:

```
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`:

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

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

2.6.3 Bind, Define and Describe Functions

Table 9-2 Bind, Define, and Describe Functions

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

2.6.4 Statement Functions

Table 9-3 Statement Functions

Function	Description
OCIStmtExecute	Execute a prepared SQL statement.
OCIStmtFetch	Fetch rows of data (deprecated).
OCIStmtFetch2	Fetch rows of data.
OCIStmtPrepare	Prepare a SQL statement.
OCIStmtPrepare2	Prepare a SQL statement.
OCIStmtRelease	Release a statement handle.

2.6.5 Transaction Functions

Table 9-4 Transaction Functions

Function	Description
OCITransCommit	Commit a transaction.
OCITransRollback	Roll back a transaction.

2.6.6 XA Functions

Table 9-5 XA Functions

Function	Description
xaoEnv	Returns OCI environment handle.
xaoSvcCtx	Returns OCI service context.

2.6.6.1 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
```

2.6.7 Date and Datetime Functions

Table 9-6 Date and Datetime Functions

Function	Description
`OCIDateAddDays`	Add or subtract a number of days.
`OCIDateAddMonths`	Add or subtract a number of months.
`OCIDateAssign`	Assign a date.
`OCIDateCheck`	Check if the given date is valid.
`OCIDateCompare`	Compare two dates.
`OCIDateDaysBetween`	Find the number of days between two dates.
`OCIDateFromText`	Convert a string to a date.
`OCIDateGetDate`	Get the date portion of a date.
`OCIDateGetTime`	Get the time portion of a date.
`OCIDateLastDay`	Get the date of the last day of the month.
`OCIDateNextDay`	Get the date of the next day.
`OCIDateSetDate`	Set the date portion of a date.

Function	Description
OCIDateSetTime	Set the time portion of a date.
OCIDateSysDate	Get the current system date and time.
OCIDateToText	Convert a date to a string.
OCIDateTimeAssign	Perform datetime assignment.
OCIDateTimeCheck	Check if the date is valid.
OCIDateTimeCompare	Compare two datetime values.
OCIDateTimeConstruct	Construct a datetime descriptor.
OCIDateTimeConvert	Convert one datetime type to another.
OCIDateTimeFromArray	Convert an array of size OCI_DT_ARRAYLEN to an OCIDateTime descriptor.
OCIDateTimeFromText	Convert the given string to Oracle datetime type in the OCIDateTime descriptor according to the specified format.
OCIDateTimeGetDate	Get the date portion of a datetime value.
OCIDateTimeGetTime	Get the time portion of a datetime value.
OCIDateTimeGetTimeZoneName	Get the time zone name portion of a datetime value.
OCIDateTimeGetTimeZoneOffset	Get the time zone (hour, minute) portion of a datetime value.
OCIDateTimeSubtract	Take two datetime values as input and return their difference as an interval.
OCIDateTimeSysTimeStamp	Get the system current date and time as a timestamp with time zone.
OCIDateTimeToArray	Convert an OCIDateTime descriptor to an array.
OCIDateTimeToText	Convert the given date to a string according to the specified format.

2.6.8 Interval Functions

Table 9-7 Interval Functions

Function	Description
OCIIntervalAdd	Adds two interval values.
OCIIntervalAssign	Copies one interval value into another interval value.
OCIIntervalCompare	Compares two interval values.
OCIIntervalGetDaySecond	Extracts days, hours, minutes, seconds and fractional seconds from an interval.
OCIIntervalSetDaySecond	Modifies days, hours, minutes, seconds and fractional seconds in an interval.
OCIIntervalGetYearMonth	Extracts year and month values from an interval.
OCIIntervalSetYearMonth	Modifies year and month values in an interval.
OCIIntervalDivide	Implements division of OCIInterval values by OCINumber values.
OCIIntervalMultiply	Implements multiplication of OCIInterval values by OCINumber values.
OCIIntervalSubtract	Subtracts one interval value from another interval value.
OCIIntervalToText	Extrapolates a character string from an interval.
OCIIntervalCheck	Verifies the validity of an interval value.
OCIIntervalToNumber	Converts an OCIInterval value into a OCINumber value.
OCIIntervalFromNumber	Converts a OCINumber value into an OCIInterval value.

Function	Description
OCIDateTimeIntervalAdd	Adds an OCIInterval value to an OCIDatetime value, resulting in an OCIDatetime value.
OCIDateTimeIntervalSub	Subtracts an OCIInterval value from an OCIDatetime value, resulting in an OCIDatetime value.
OCIIntervalFromText	Converts a text string into an interval.
OCIIntervalFromTZ	Converts a time zone specification into an interval value.

2.6.9 Number Functions

Table 9-8 Number Functions

Function	Description
OCINumberAbs	Compute the absolute value.
OCINumberAdd	Adds NUMBERs.
OCINumberArcCos	Compute the arc cosine.
OCINumberArcSin	Compute the arc sine.
OCINumberArcTan	Compute the arc tangent.
OCINumberArcTan2	Compute the arc tangent of two NUMBERs.
OCINumberAssign	Assign one NUMBER to another.
OCINumberCeil	Compute the ceiling of NUMBER.
OCINumberCmp	Compare NUMBERs.
OCINumberCos	Compute the cosine.
OCINumberDec	Decrement a NUMBER.
OCINumberDiv	Divide two NUMBERs.
OCINumberExp	Raise e to the specified NUMBER power.
OCINumberFloor	Compute the floor of a NUMBER.
OCINumberFromInt	Convert an integer to an Oracle NUMBER.
OCINumberFromReal	Convert a real to an Oracle NUMBER.
OCINumberFromText	Convert a string to an Oracle NUMBER.
OCINumberHypCos	Compute the hyperbolic cosine.
OCINumberHypSin	Compute the hyperbolic sine.
OCINumberHypTan	Compute the hyperbolic tangent.
OCINumberInc	Increments a NUMBER.
OCINumberIntPower	Raise a given base to an integer power.
OCINumberIsInt	Test if a NUMBER is an integer.
OCINumberIsZero	Test if a NUMBER is zero.
OCINumberLn	Compute the natural logarithm.
OCINumberLog	Compute the logarithm to an arbitrary base.
OCINumberMod	Modulo division.
OCINumberMul	Multiply NUMBERs.
OCINumberNeg	Negate a NUMBER.
OCINumberPower	Exponentiation to base e.
OCINumberPrec	Round a NUMBER to a specified number of decimal places.
OCINumberRound	Round a NUMBER to a specified decimal place.
OCINumberSetPi	Initialize a NUMBER to Pi.

Function	Description
OCINumberSetZero	Initialize a NUMBER to zero.
OCINumberShift	Multiply by 10, shifting specified number of decimal places.
OCINumberSign	Obtain the sign of a NUMBER.
OCINumberSin	Compute the sine.
OCINumberSqrt	Compute the square root of a NUMBER.
OCINumberSub	Subtract NUMBERs.
OCINumberTan	Compute the tangent.
OCINumberToInt	Convert a NUMBER to an integer.
OCINumberToReal	Convert a NUMBER to a real.
OCINumberToRealArray	Convert an array of NUMBER to a real array.
OCINumberToText	Converts a NUMBER to a string.
OCINumberTrunc	Truncate a NUMBER at a specified decimal place.

2.6.10 String Functions

Table 9-9 String Functions

Function	Description
OCIStringAllocSize	Get allocated size of string memory in bytes.
OCIStringAssign	Assign string to a string.
OCIStringAssignText	Assign text string to a string.
OCIStringPtr	Get string pointer.
OCIStringResize	Resize string memory.
OCIStringSize	Get string size.

2.6.11 Cartridge Services and File I/O Interface Functions

Table 9-10 Cartridge Services and File I/O Interface Functions

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

2.6.12 LOB Functions

Table 9-11 LOB Functions

Function	Description
OCILobRead	Returns a LOB value (or a portion of a LOB value).
OCILOBWriteAppend	Adds data to a LOB value.
OCILobGetLength	Returns the length of a LOB value.
OCILobTrim	Trims data from the end of a LOB value.
OCILobOpen	Opens a LOB value for use by other LOB functions.
OCILobClose	Closes a LOB value.

2.6.13 Miscellaneous Functions

Table 9-12 Miscellaneous Functions

Function	Description
OCIClientVersion	Return client library version.
OCIErrorGet	Return error message.
OCIPGErrorGet	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 bufsiz, ub4 type)`
OCIPasswordChange	Change password.
OCIPing	Confirm that the connection and server are active.
OCIServerVersion	Get the Oracle version string.

2.6.14 Supported Data Types

Table 9-13 Supported Data Types

Function	Description
ANSI_DATE	ANSI date
SQLT_AFC	ANSI fixed character
SQLT_AVC	ANSI variable character
SQLT_BDOUBLE	Binary double
SQLT_BIN	Binary data
SQLT_BFLOAT	Binary float
SQLT_CHR	Character string
SQLT_DAT	Oracle date
SQLT_DATE	ANSI date
SQLT_FLT	Float
SQLT_INT	Integer
SQLT_LBI	Long binary
SQLT_LNG	Long
SQLT_LVB	Longer long binary
SQLT_LVC	Longer longs (character)
SQLT_NUM	Oracle numeric
SQLT_ODT	OCI date type

Function	Description
SQLT_STR	Zero-terminated string
SQLT_TIMESTAMP	Timestamp
SQLT_TIMESTAMP_TZ	Timestamp with time zone
SQLT_TIMESTAMP_LTZ	Timestamp with local time zone
SQLT_UIN	Unsigned integer
SQLT_VBI	VCS format binary
SQLT_VCS	Variable character
SQLT_VNU	Number with preceding length byte
SQLT_VST	OCI string type

2.7 OCI Error Codes – Reference

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

Error Code	Condition Name	Oracle Error Code
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
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

Error Code	Condition Name	Oracle Error Code
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

For more information about Postgres error codes, please see the PostgreSQL core documentation at:

https://www.postgresql.org/docs/10/static/errcodes-appendix.html