Introduction to 4D for OCI

By Yvan Ayaay, Technical Support Engineer, 4D Inc.
TN 05-39

Introduction

The 4D for OCI plug-in allows 4D to communicate with Oracle Servers. It uses the Oracle Call Interface (OCI), an Application Program Interface (API), which allows native access to Oracle Servers. It makes it possible to access data from an Oracle database within any 4th Dimension or 4D Server based applications acting as Oracle clients. In this technical note, the basic concepts and some of the basic operations of 4D for OCI will be discussed.

Overview

The Oracle Call Interface (OCI) is an application programming interface (API) that allows applications to interact with one or more Oracle Servers. OCI provides a library of standard database access and retrieval functions in the form of a dynamic runtime library. This OCI library needs to be installed for the 4D for OCI plug-in to work. It provides native connectivity to Oracle database servers and gives programs the capability to perform different database operations on an Oracle database. The OCI library allocates handles that stores context information, connection information, bind information, data information, and error information. An OCI application can then access specific information contained in these handles. It controls connection, executes SQL statements, and stores and processes result rows of queries.

Installation and Configuration

4D for OCI provides support for Oracle 8.16, Oracle 9 and Oracle 10 databases. In order for it to function, the files (libraries) needed to run OCI must be installed. You can install the libraries by downloading and installing the Oracle Database Client Software from Oracle Technology Network Web Site (http://www.oracle.com/technology/software/). Navigate to each of the download pages for the products that you want to install. For instance, you can choose to select the Oracle Database 10g download link and select the one for your operating system.

OCI Library Installation for Windows

For installation of the Oracle Client in Windows, you can simply use the Oracle Universal Installer. Once you downloaded the Oracle Client installer and once the file is extracted, click on the Setup.exe file to launch the Oracle Universal Installer which guides you through the installation process (the Administrator type was selected as the installation type on the setup here). Here's a link to the installation guide:

http://download-east.oracle.com/docs/cd/B19306_01/install.102/b14312/install.htm#BABJGGJH

The OCI Library comes installed with the Oracle Client application. After the installation, the connection to the Oracle database should be properly configured by setting up the *tnsnames.ora* file. You can create this file and configure it using the Net Configuration Assistant program that gets installed together with the Oracle Database Client. Launch this application from the Start Menu->Oracle-Home_Name->Configuration->Migration Tools. Then, configure the connection by entering the name of the Oracle database you want to connect to, the protocol, and the IP Address of the database server. The *tnsnames.ora* is created at the following location:

ORACLE_BASE\ORACLE_HOME\network\admin\tnsnames.ora

You can manually create it using a text editor with proper configuration values and put it at the above location. Each entry follows the syntax as shown below:

```
# tnsnames.ora Network Configuration File:
C:\oracle\product\10.2.0\client_1\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.

ORACLE4D =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 10.64.0.20)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = oracle4d)
    )
  )
```

Once the client application is installed, you can now use the 4D for OCI.

OCI Library Installation for Mac OS 10.3.x

To install the OCI Library on Mac OS 10 (tested on Mac OS 10.3.9 machine), the Oracle 10g Instant Client for Mac OS X can be used. You can download the Instant Client at this location:
http://www.oracle.com/technology/software/tech/oci/instantclient/htdocs/macsoft.html
The file you want is called "instantclient-basic-macosx-10.1.0.3.zip".

Here are the steps that were performed to install the Instant Client and make it work with 4D for OCI:

NOTE: All Terminal commands are CASE SENSITIVE!

1) Install the Oracle 10g Instant Client files to the proper location
To get the Instant Client working with 4D for OCI the files must be in a directory called "/Oracle". Additionally you need the files to be owned by "root". Here are the steps to place the files in the proper location:
-Extract the "instantclient-basic-macosx-10.1.0.3.zip" archive to your desktop. This will create a folder called "instantclient10_1".
-Open a Terminal window.
-Execute the command 'sudo sh'. This opens an sh shell as root.
-Enter your password when prompted. If you can not complete this step stop, you need 'sudo' access to do this install.
-Execute the command 'mkdir /Oracle'.
-Type 'cp ' and then drag the "instantclient10_1" folder from your desktop to the Terminal window. **Do not** press return.
-Type backspace, then '/* /Oracle'. At this point the command should look like:
sh-2.05b# cp <your user home>/Desktop/instantclient10_1/* /Oracle
-Press return. This will copy all of the Instant Client files to the "/Oracle" folder.
-Keep the terminal window open and move to step 2.

2) Copy the required Oracle library to the proper location.
-Execute the command 'cp /Oracle/libclntsh.dylib.10.1 /usr/lib/libclntsh.dylib'.
-Execute the command 'chmod 777 /usr/lib/libcIntsh.dylib'.
-Execute the command 'exit'. This takes you out of "root" mode.
-Keep the terminal window open and move to step 3.

3) Create/Update an "environment.plist" file for the user that will be using 4D for OCI.
-Execute the command 'cd ~'. This takes you to your home directory.
-Execute the command 'mkdir .MacOSX'. Note the '.' is required.

You now need to create a file called "environment.plist". This is a Macintosh properties file and is a plain text, XML file. How you create it will vary depending on how your machine is set up. E.g. there is a Property List Editor in Mac OS X that you can use. You can also use a plain text editor, or 'vi' from the Terminal. However when you create the file, the contents should be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>DYLD_LIBRARY_PATH</key>
        <string>/Oracle</string>
        <key>ORACLE_HOME</key>
        <string>/Oracle/OCI/oci</string>
        <key>ORA_NLS33</key>
        <string>/Oracle/OCI/oci/ocommon/nls/admin/data</string>
        <key>TNS_ADMIN</key>
        <string>/Oracle/OCI/oci/network/admin</string>
</dict>
</plist>
```

If you did not create this file in the ".MacOSX" folder, move it there. You may need to do this from a Terminal as, by default, Finder does not display folders that start with a '.' (these are hidden folders in *nix). To copy the file from a Terminal:

```
cp <folder that contains>/environment.plist ~/.MacOSX
```

Once the file is in place, log out and log back in.

Note:
If you do not want to install the Instant Client files to /Oracle you must use the 4D for OCI command "OCISetEnv" to set the values for "ORACLE_HOME", "ORA_NLS33" and "TNS_ADMIN". The 4D for OCI plug-in uses these values as defaults (as seen in "environment.plist"):

ORACLE_HOME=/Oracle/OCI/oci
ORA_NLS33=/Oracle/OCI/oci/ocommon/nls/admin/data
TNS_ADMIN=/Oracle/OCI/oci/network/admin

Just like in the Windows setup, the TNSNAMES.ORA file should be created and properly setup to set the target IP address of the oracle machine. Here are the steps to do this:
1. Create a TNSNAMES.ORA file using a text editor and type entries as shown below.

```
oracle4d =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL =TCP)(HOST = 10.96.0.61)(POST = 1521))
```

```
  )
  (CONNECT_DATA =
    (SERVICE_NAME = oracle4d)
  )
)
```

Make sure you have the correct IP address for your Oracle server provided in the HOST parameter.

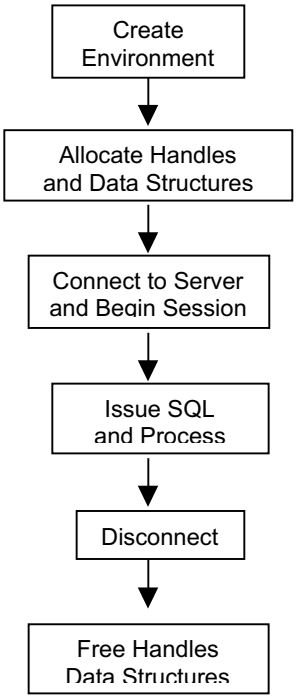
2. Open your terminal window and copy the file to /private/etc directory.

You should now be able to use the 4D for OCI plug-in.

Note: If the 4D for OCI plug-in shows as disabled in the Explorer window->Components section in the Design environment, it is possible that the OCI Library is not installed properly.

Basic OCI Program Structure

An OCI application follows a program structure as shown below. This program flow should be followed when developing an OCI application within 4D.



Initially, an OCI environment is created and the handles are allocated. Then, a connection to the server is made and a session is started. In this session, SQL statement are issued and processed with changes applied to database

through the process of commit. Lastly, disconnecting from the server and freeing of handles are performed.

Below are the steps in developing an OCI application within 4D (in the Basic Operation section below, all these steps are put together in the examples):

Step 1: *Creation and initialization of the OCI programming environment and threads.*

An OCI environment needs to be created and initialized at the beginning of an OCI application. All OCI functions are executed in the context of this environment. The 4D for OCI command OCIEnvCreate() needs to be invoked before executing other OCI calls. This command creates the OCI environment and initializes its handle. This handle is then used in allocating other handle types. Below is a code example on how to use this command.

```
$error_l:=OCIEnvCreate   ($Envhp;OCI_HTYPE_ENV  )
```

Step 2: *Allocate handles and descriptors.*

Most OCI calls use handles as parameter(s). A handle is basically a pointer to a storage area allocated by the OCI library. Information about the context, connection, data, and OCI function calls are stored in handles that are accessed by OCI applications. Thus, the necessary handles need to be allocated in an OCI program. Among these handles that need to be allocated are the service context handle, server handle, session handle, statement handle, and error handle.

The *service context handle* defines attributes that determine the operational context for OCI calls to a server. It comprises of three additional handles: server, session, and transaction handles. The *server handle* identifies connection to the database server. The *session handle* identifies user's roles and privileges, and the operational context. And the *transaction handle* defines transaction in which the SQL operations are performed.
The *statement handle* defines SQL Statements and its associated attributes.
The *error handle* which is passed as parameter to an OCI call contains information about errors that occurred in an OCI operation. Information about the error can be checked using the OCIErrorGet() command.
Below is a sample code that shows allocation of these handles:

```
$error_l:=OCIHandleAlloc   ($Envhp;$svchp;OCI_HTYPE_SVCCTX  ) `Service context handle
$error_l:=OCIHandleAlloc   ($Envhp;$authp;OCI_HTYPE_SESSION  ) `  Session handle
$error_l:=OCIHandleAlloc   ($Envhp;$srvhp;OCI_HTYPE_SERVER  ) `  Server handle
$error_l:=OCIHandleAlloc   ($Envhp;$Stmthp;OCI_HTYPE_STMT  ) `Statement  Handle
$error_l:=OCIHandleAlloc   ($Envhp;$errhp;OCI_HTYPE_ERROR  ) `  Error handle
```

Step 3: *Establish server connections and user sessions.*

Once the OCI environment is initialized and the necessary handles allocated, the next step is to establish server connections and to begin user

sessions. There are two options in doing this: Single User / Connection; or Multiple Sessions / Connections.

If only a single user session for each database connection is required, a simplified logon call OCILogon() can be used. As shown in the code below, the environment, error, and service context handle are passed together with the database access parameters.

```
$error_l:=OCILogon   ($Envhp;$errhp;$svchp;"scott";"tiger";"oracle4d")
```

For multiple users' sessions, different sets of OCI calls are performed: calls to attach to the server and to start a session. As shown below, the command OCIServerAttach() is executed to create an access path to the server and, then, the server attributes are set. With the command OCISessionBegin, a session for the user is established.

```
$error_l:=OCIServerAttach     ($srvhp;$errhp;"oracle4d")
$error_l:=OCIAttrSetVal    ($svchp;$srvhp;OCI_ATTR_SERVER  ;$errhp)
$error_l:=OCIAttrSetText     ($authp;"scott";22;$errhp)
$error_l:=OCIAttrSetText    ($authp;"tiger";23;$errhp)
$error_l:=OCISessionBegin    ($svchp;$errhp;$authp;1;0)
$error_l:=OCIAttrSetVal    ($svchp;$authp;OCI_ATTR_SESSION  ;$errhp)
```

Step 4: *Issue SQL and Process Data*

Once a connection to the server is established and a user session is started, it is now possible to manipulate and access data from the database server by executing SQL statements on the server. You can either change data in the oracle database table or retrieve data from it into an output variable such as an array, variable, field etc. To change data in the database by using DML (Data Manipulation Language) commands like insert, update, and delete, a binding of input variables should be performed before executing the statement. On the other hand, to retrieve data from database using SQL statements, defining of output variables are performed. In both cases, preparing of the SQL statement is needed. The statement OCIStmtPrepare() as shown below should be executed before the binding or defining.

```
$error_l:=OCIStmtPrepare    ($Stmthp;$Errhp;$sql_t;OCI_DEFAULT   )
```

You can manipulate data in the oracle database by ***binding*** (changing data in the database) or ***defining*** (retrieving data from the Oracle database into 4D).

Binding: Changing data in the database.

When passing data to Oracle as part of the SQL statement to perform, you will need to bind associated place holders in the statement to input variables in 4D. For example, when you perform an insert, you pass the values in your variables in 4D to Oracle through binding. For DML statements and queries

with input variables, binding can be performed by using OCIBindByPos() or by using OCIBindByName() to bind the address of each input variable or array to each placeholder in the statement.
For example in the statement "INSERT INTO DEPT (DEPTNO,DNAME, LOC) VALUES (:DEPTNO,:DName,:LOC)", you can bind values assigned to 4D variables deptno, dname, and location just like the code below:

```
myptrvar1:=Get pointer("deptno")
myptrvar2:=Get pointer("dname")
myptrvar3:=Get pointer("location")
$error_l:=OCIBindByPos ($Stmthp;$dpp;$Errhp;1;myptrvar1;SQLT_INT
;tindp;trlenp;trcodep;OCI_DEFAULT )
$error_l:=OCIBindByPos ($Stmthp;$dpp;$Errhp;2;myptrvar2;SQLT_STR
;tindp;trlenp;trcodep;OCI_DEFAULT )
$error_l:=OCIBindByPos ($Stmthp;$dpp;$Errhp;3;myptrvar3;SQLT_STR
;tindp;trlenp;trcodep;OCI_DEFAULT )
```

After binding, the statement is then executed using the OCIStmtExecute() command.

Defining: Retrieving data from an Oracle database into 4D.

When you perform query statements, data is returned from the Oracle database to 4D. When processing a query, output variables within 4D should be defined. Defining of output variables associates where the returned results are stored and what type of data format. For instance, if you have SQL Statement that gets the job description of all employees in the Employee table (Select job from Employee), you can perform a code just like below to define the output variable (an array as) in 4D:

```
` SQL Statement: Select job from Employee
ARRAY TEXT(Returnvalue_at;0)
$error_l:=OCIDefineByPos ($Stmthp;$define;$Errhp;1;->vReturnvalue_at;SQLT_STR
;tindp;trlenp;trcodep;OCI_DEFAULT
```

When the statement is executed, the result needs to be fetched--it will be stored in the text array Returnvalue_at. The 4th parameter of the OCIDefineByPos command is the column number (1 in the example for first column which is job) of the SQL Statement that the output variable is assigned. To fetch results, you use the OCIStmtFetch().

The code example below shows this:

```
$error_l:=OCIStmtPrepare ($Stmthp;$Errhp;$sql_t;OCI_DEFAULT ) `Length($sql_t)) `  ***
Prepare SQL statement
If ($error_l=0)

   $error_l:=OCIDefineByPos($Stmthp;$define;$Errhp;1;>vReturnvalue_t;SQLT_STR;tindp;trlenp
   ;trcodep;OCI_DEFAULT )
```

```
    ` *** Bind Oracle variable by position toward 4D
  If ($error_l=0)
    $error_l:=OCIStmtExecute    ($svchp;$Stmthp;$Errhp;0;0;0;0;OCI_DEFAULT   )
      ` *** Execute the SQL statment
      If If($error_l=0)
      Repeat
        $error_l:=OCIStmtFetch    ($Stmthp;$Errhp;1) `  *** Fetch the result 1 row at the
time
        If ($error_l=0)
          APPEND TO ARRAY(Returnvalue_at;vReturnvalue_t)
        End if
      Until ($error_l=OCI_NO_DATA )
```

Step 5: *Commit*
For the changes in the database to take effect, changes need to be committed using the OCITransCommit()command. The service context handle is passed as parameter as shown below. If the database is terminated without commit, the changes will not take affect as an automatic rollback is performed.

```
$error_l:=OCITransCommit    ($svchp;$Errhp;OCI_DEFAULT   )
```

Step 6: *Terminate user sessions and server connections.*
Before terminating the application the user sessions and server connections should be terminated. The user session and server connection are ended by calling the OCISessionEnd() and OCIServerDetach() respectively.

Step 7: *Free handles.*

An OCI application must free all handles when they are no longer needed. The OCIHandleFree() function frees handles.

OCI Basic Operations

In this section, some of the basic operations supported by OCI will be shown. The first example will retrieve data from an Oracle database and will fetch the results into an array. The second example inserts data into an Oracle database.

Example 1: Retrieving data from an Oracle database into 4D using SQL.

```
`Method: QueryData
`Description: This method will retrieved data from Oracle database and place the result
`                into an array.
```

```
C_LONGINT($error_l)
C_LONGINT($Envhp;$errhp;$svchp;$authp;$srvhp)
C_LONGINT($srvhp;$svchp;$authp;$errhp)

ARRAY TEXT(Returnvalue_at;0)
C_TEXT(vReturnvalue_t;null_t)
C_POINTER(tindp;trlenp;trcodep)
tindp:=->null_t ` *** Provide a null value to the text variable

C_TEXT($sql_t)

 ` Select statement.
$sql_t:="Select job from emp"

 ` *** Create the OCI Environment
$error_l:=OCIEnvCreate ($Envhp;OCI_HTYPE_ENV )

If ($error_l=0)

   `*** Allocate Handles and Data Structures
  $error_l:=OCIHandleAlloc ($Envhp;$errhp;OCI_HTYPE_ERROR ) ` Allocate Error handle.
  If ($error_l=0)
    $error_l:=OCIHandleAlloc ($Envhp;$svchp;OCI_HTYPE_SVCCTX ) `Allocate Service
Context Handle.
    If ($error_l=0)
      $error_l:=OCIHandleAlloc ($Envhp;$authp;OCI_HTYPE_SESSION )
        ` Allocate User Session Handle
      If ($error_l=0)
        $error_l:=OCIHandleAlloc ($Envhp;$srvhp;OCI_HTYPE_SERVER )
          `Allocate Server session handle
        If ($error_l=0)
          $loginok_b:=True
        End if
      End if
    End if
  End if
End if

 ` *** Connect to Server
$define:=0
If ($error_l=0)
  $error_l:=OCILogon ($Envhp;$errhp;$svchp;"scott";"tiger";"oracle4d") ` connect server
using simple logon call.

   `**** Issue SQL Statement
  If ($error_l=0)
    $error_l:=OCIHandleAlloc ($Envhp;$Stmthp;OCI_HTYPE_STMT ) ` *** Create SQL
Statement Handler
    If ($error_l=0)
      $error_l:=OCIStmtPrepare ($Stmthp;$Errhp;$sql_t;Length($sql_t)) ` *** Prepare SQL
statement
      If ($error_l=0)
```

```
        $error_l:=OCIDefineByPos  ($Stmthp;$define;$Errhp;1;->vReturnvalue_t;SQLT_STR
;tindp;trlenp;trcodep;OCI_DEFAULT ) `  *** Define output variables
        If ($error_l=0)
          $error_l:=OCIStmtExecute  ($svchp;$Stmthp;$Errhp;0;0;0;0;OCI_DEFAULT ) `  ***
Execute the SQL statment
          If ($error_l=0)
            Repeat
              $error_l:=OCIStmtFetch  ($Stmthp;$Errhp;1) `  *** Fetch the result 1 row at
the time
              If ($error_l=0)
                APPEND TO ARRAY(Returnvalue_at;vReturnvalue_t) `  *** Put fetch results
into array.
              End if
            Until ($error_l=OCI_NO_DATA )

          End if
        End if

        $error_l:=OCIHandleFree  ($Stmthp)

      End if
    End if
  End if
End if



If ($loginok_b)
    `***Disconnect from server
  $error_l:=OCILogoff  ($svchp;$errhp)

   `*****  Free Handles
  $error_l:=OCIHandleFree  ($svchp)
  $error_l:=OCIHandleFree  ($srvhp)
  $error_l:=OCIHandleFree  ($errhp)
  $error_l:=OCIHandleFree  ($authp)
End if
```

Example 2. Inserting data into an Oracle database.

```
`Method: InsertData
`Description: This method will insert a record in the Oracle database.

C_LONGINT($error_l)
C_LONGINT($Envhp;$errhp;$svchp;$authp;$srvhp)
C_LONGINT($srvhp;$svchp;$authp;$errhp)

C_TEXT(vReturnvalue_t;null_t)
C_POINTER(tindp;trlenp;trcodep)
C_POINTER(myptrvar1;myptrvar2;myptrvar3)
```

```
C_LONGINT(deptno1)
C_TEXT(dname1;loc1)
C_TEXT($sql_t)


deptno1:=12
dname1:="Management"
loc1:="Seattle"

myptrvar1:=Get pointer("deptno1")
myptrvar2:=Get pointer("dname1")
myptrvar3:=Get pointer("loc1")

tindp:=->null_t ` *** Provide a null value to the text variable

 ` SQL STATEMENT to insert record to DEPT table in Oracle.
  `The variables preceded by : are place holders.
$sql_t:="INSERT INTO DEPT (DEPTNO,DNAME, LOC) VALUES (:DEPTNO,:DName,:LOC)"


 ` *** Create the OCI Environment
$error_l:=OCIEnvCreate ($Envhp;OCI_HTYPE_ENV )
If ($error_l=0)

   `*** Allocate Handles and Data Structures
  $error_l:=OCIHandleAlloc ($Envhp;$errhp;OCI_HTYPE_ERROR )
     ` Allocate Error handle.
  If ($error_l=0)
    $error_l:=OCIHandleAlloc ($Envhp;$svchp;OCI_HTYPE_SVCCTX )
     `Allocate Service Context Handle.
    If ($error_l=0)
      $error_l:=OCIHandleAlloc ($Envhp;$authp;OCI_HTYPE_SESSION )
     ` Allocate User Session Handle
      If ($error_l=0)
        $error_l:=OCIHandleAlloc ($Envhp;$srvhp;OCI_HTYPE_SERVER )
       `Allocate Server session handle
        If ($error_l=0)
          $loginok_b:=True
        End if
      End if
    End if
  End if
End if

 ` *** Connect to Server and Begin Session
$define:=0
If ($error_l=0)
  $error_l:=OCIServerAttach ($srvhp;$errhp;"oracle4d")
    `Create access path to oracle database
  If ($error_l=0)
    $error_l:=OCIAttrSetVal ($svchp;$srvhp;OCI_ATTR_SERVER ;$errhp)
     ` Set server attributes
    If ($error_l=0)
```

```
      $error_l:=OCIAttrSetText    ($authp;"scott";22;$errhp) `Set    user name to access
server. 22 is attribute type.
      If ($error_l=0)
        $error_l:=OCIAttrSetText    ($authp;"tiger";23;$errhp)    ` Set password. 23 is attribute
type.
        If ($error_l=0)
          $error_l:=OCISessionBegin    ($svchp;$errhp;$authp;1;0)    ` Begin user session
          If ($error_l=0)
            $error_l:=OCIAttrSetVal    ($svchp;$authp;OCI_ATTR_SESSION  ;$errhp)  `Set
session attribute


             `****    Issue SQL Statement
           If ($error_l=0)
             $error_l:=OCIHandleAlloc    ($Envhp;$Stmthp;OCI_HTYPE_STMT  )
             ` *** Create SQL Statement Handler
             If ($error_l=0)
               $error_l:=OCIStmtPrepare    ($Stmthp;$Errhp;$sql_t;Length($sql_t))
                 ` *** Prepare SQL statement
               If ($error_l=0)
                  `  ***Bind  variables to positions in the SQL statement.
                   `The 4th parameter is the position in the SQL Statment, the 5th parameter is
                   `       a pointer to the variable to bind, and the 6th parameter is the data type.
                 $error_l:=OCIBindByPos    ($Stmthp;$dpp;$Errhp;1;myptrvar1;SQLT_INT
;tindp;trlenp;trcodep;OCI_DEFAULT )
                 $error_l:=OCIBindByPos    ($Stmthp;$dpp;$Errhp;2;myptrvar2;SQLT_STR
;tindp;trlenp;trcodep;OCI_DEFAULT )
                 $error_l:=OCIBindByPos    ($Stmthp;$dpp;$Errhp;3;myptrvar3;SQLT_STR
;tindp;trlenp;trcodep;OCI_DEFAULT )
                 If  ($error_l=0)
                   $error_l:=OCIStmtExecute    ($svchp;$Stmthp;$Errhp;1;0;0;OCI_DEFAULT  )
             ` *** Execute the SQL statment
                    $error_l:=OCITransCommit    ($svchp;$Errhp;OCI_DEFAULT  )
           `***   Commit changes to database.
                 End if
                   `***    Free statement  handle.
                 $error_l:=OCIHandleFree    ($Stmthp)
               End if

             End if
           End if
         End if
       End if
     End if
   End if
End if

If ($loginok_b)
    `***Disconnect  from server
     $error_l:=OCISessionEnd    ($svchp;$errhp;$authp)
  $error_l:=OCIServerDetach    ($srvhp;$errhp)
```

```
   `*****  Free Handles
  $error_l:=OCIHandleFree  ($svchp)
  $error_l:=OCIHandleFree  ($srvhp)
  $error_l:=OCIHandleFree  ($errhp)
  $error_l:=OCIHandleFree  ($authp)
End if
```

Summary

The 4D for OCI plug-in allows you to access and manipulate data from an Oracle Database through the OCI(Oracle Call Interface) API. The OCI Library must first be installed before you can use 4D for OCI and the tnsnames.ora file should be configured for the server. The OCI Library performs the native access to the Oracle database. An OCI program structure is followed to develop an OCI application within 4D. The OCI program flow follows this order: create environment, allocate handles, connect to server and begin session, process SQL statement, disconnect, and free handles. 4D for OCI plug-in makes it possible to communicate to an Oracle database from 4D.