OCI Mapper 2004-3

By Josh Fletcher, Technical Support Engineer, 4D, Inc.

Technical Note 06-06

Abstract

This Technical Note accompanies an updated version of the OCI Mapper component, which was introduced in Technical Note 03-44, "Migrating from 4D for Oracle to 4D for OCI using OCI Mapper". The main focus of this Technical Note is to summarize the updates that have been made to the OCI Mapper and provide more insight into 4D for OCI programming, both on its own, and in comparison to 4D Oracle. Installation procedures for the OCI Mapper component are included.

This Technical Note is the first part of a two-part OCI Mapper update. In the next part of the release a debug version of the OCI Mapper will be provided. This version will feature the logging of OCI Mapper methods as well as 4D for OCI calls to a text file in order to facilitate advanced analysis of 4D for OCI programming problems. The reason for the two-part release is the debugging code tends to make the OCI Mapper methods harder to read.

Introduction

The focus of this Technical Note (and the OCI Mapper component) is two-fold:

- For 4D Oracle databases the OCI Mapper provides a transition from 4D Oracle to 4D for OCI without modification of the source code.
- For developers who are new to 4D for OCI (or OCI programming in general) the OCI Mapper demonstrates how to implement a framework that provides higher-level commands than those found in 4D for OCI.

4D Oracle developers should find useful comparisons between 4D Oracle commands and 4D for OCI commands in this Technical Note and a robust introduction to 4D for OCI programming via the OCI Mapper component.

4D for OCI developers should find the OCI Mapper useful as a demonstration of how to implement a high-level, API-like framework for 4D for OCI development.

In updating the OCI Mapper an attempt has been to both fix bugs in the original design as well as point out highlights of the design, how it differs from 4D Oracle, and what areas might need improvement.

Please note that the OCI Mapper component is provided as "open source". The developer is free to modify the OCI Mapper code at will and, in fact, is encouraged to do so.

It should be noted that the majority of the modifications made to the OCI Mapper have been documented in the code. That is, the original code is retained (commented) along with the new code to demonstrate why the changes were made. In some cases these are simple bug fixes, but in many cases a change might highlight a particular hurdle to OCI programming. In other words, the old code is retained as a demonstration of what **not** to do when you are developing with 4D for OCI.

Also note that, in an attempt to not cover previously addressed topics, performance comparisons between 4D Oracle and 4D for OCI are not covered in this Technical Note. Please see Technical Note 03-44, "Migrating from 4D for Oracle to 4D for OCI using OCI Mapper" if you would like to see some benchmarking comparisons of the two plug-ins.

Overview of 4D Oracle and 4D for OCI

Adapted from Technical Note 03-44, "Migrating from 4D for Oracle to 4D for OCI using OCI Mapper"

Both 4D Oracle and 4D for OCI are connectivity plug-ins. They provide a set of 4th Dimension external routines that allow 4th Dimension to communication with Oracle databases. Their basic functionalities are to display, manipulate, and modify data stored in an Oracle database. Combining these plug-ins with 4th Dimension's rapid user-interface development makes for a great environment to create front-end applications for Oracle databases.

However, the two plug-ins differ in their level of support for 4D and Oracle development:

Supports	4D for Oracle	4D for OCI
4D 2004	No	Yes
High-Level Commands	Yes	No
Low-Level Commands	Some	Yes
OCI Driver/Oracle Client	Version 7.x or 8.x	Version 8.1.6 or higher

Notice that, in order to continue Oracle development in 4D 2004, 4D for OCI must be used. At the same time, the high-level commands that existed 4D Oracle are gone.

Advantage of using 4D for OCI over 4D Oracle

Why use 4D for OCI instead of 4D Oracle?

The simplest answer is, of course, that 4D Oracle no longer exists for the most recent version of 4th Dimension (2004). However, there are better reasons as well.

4D Oracle provided many high and low-level commands, which allowed the developer to accomplish many different tasks. At the same time the design of 4D

Oracle was more restrictive and did not allow the developer to alter the way the commands worked.

On the other hand 4D for OCI provides extremely low-level commands, allowing the developer to implement customized solutions in their database. Because 4D for OCI is flexible and the commands are low-level the developer is in control of their Oracle programming.

It was also shown in Technical Note 03-44 that 4D for OCI code executes much faster than 4D Oracle.

Where does the OCI Mapper fit in?

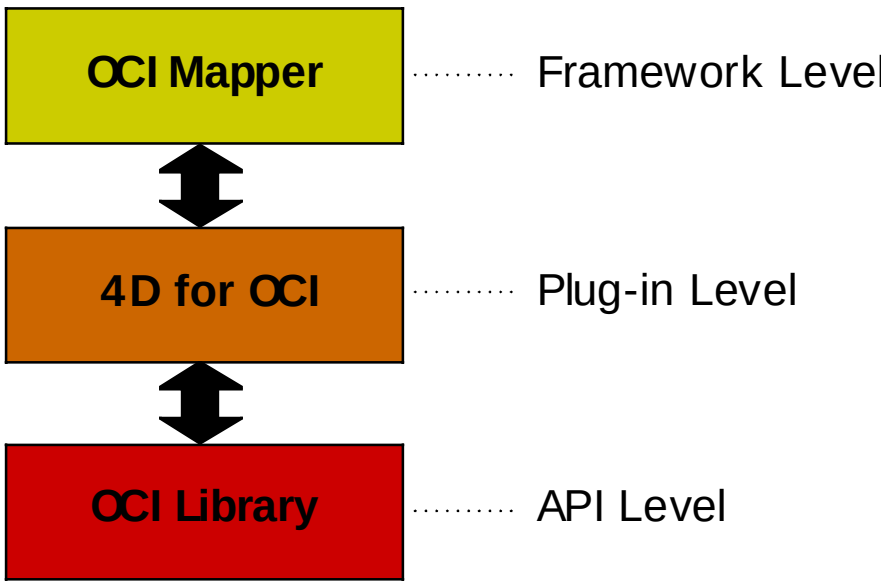
The drawback to using 4D for OCI is also its strength: low-level commands. Developing a database using only low-level 4D for OCI commands represents a tremendous amount of coding. It is desirable to "wrap" the 4D for OCI commands into an API-like framework that fits the target design. The OCI Mapper is one such framework.

Introduction to the OCI Mapper

The OCI Mapper is a framework that emulates the high-level commands found in 4D Oracle. It is provided in the form of a 4D component (including source code). Its methods were implemented with the native 4D language and 4D for OCI commands.

Architecture

Here is how the OCI Mapper fits into the Oracle development picture:



In the above diagram the framework would be provided by the developer, the plug-in is provided by 4D, and the API is provided by Oracle.

OCI Mapper 2004-3 in-depth

The OCI Mapper 2004-3 provides a host of fixes and improvements to the original design. In this section you will find highlights from this update as well as important information for 4D Oracle and 4D for OCI developers.

4D Oracle Developers

The main purpose of OCI Mapper for 4D Oracle developers is to provide an upgrade path from 4D 2003 to 4D 2004, and thus 4D Oracle to 4D for OCI, that does not require modification of the source code in the database. The OCI Mapper component uses the same command names as 4D Oracle, and emulates the behavior of those commands. The idea is to allow the older 4D Oracle database to be up and running with 4D for OCI as quickly as possible.

The secondary purpose of the OCI Mapper for 4D Oracle developers is to demonstrate how 4D for OCI programming differs from 4D Oracle. Thus it is advisable not to treat the OCI Mapper as a solution, but rather as a learning tool.

4D for OCI Developers

The main purpose of OCI Mapper for 4D for OCI developers is to provide an example of how to implement a framework that wraps 4D for OCI commands into useful, high-level commands.

Again, it is advisable not to treat the OCI Mapper as a solution, but rather as a learning tool.

Important Notes For OCI Mapper 2004-3

This section highlights important issues with the OCI Mapper.

Concurrent Cursors

While the OCI Mapper does provide support for multiple, concurrent cursors (via arrays to track the state of the cursors) multiple sets of output parameters are not supported.

For example, given **cursor1** and **cursor2**:

- Define output parameters for **cursor1**.
- Define output parameters for **cursor2**.
- Execute **cursor1**.

- Fetch results.

In this case the results are fetched into the output parameters for **cursor2**, not **cursor1**. Instead of doing this, do:

- Define output variables for **cursor1**.
- Execute **cursor1**.
- Fetch results.
- Close **cursor1**.
- Repeat for **cursor2**.

Alternatively the developer is free to modify the OCI Mapper to support multiple sets of output parameters.

Error Handling

There are instances in the OCI Mapper code where 4D for OCI calls are executed but there is no subsequent error handling block. However in these cases it is important to understand that, were an error to have occurred, something is most likely fundamentally at fault with the design of the OCI Mapper. This is as opposed to a situation where error handling is needed to address "normal" behavior and branch code accordingly.

For debugging purposes it is often helpful to install a custom error handler using OD ON ERROR CALL (the CheckError method may be used). This way, anytime a 4D for OCI error occurs, some form of error handling will be in place. Additionally, as mentioned before, a version of the OCI Mapper with debug logging will be made available.

Of course the developer is free to modify the OCI Mapper to implement greater error-handling coverage.

LOBs

The OCI Mapper does not currently support Large Object (LOB) types. LOBs are BLOB and Picture in 4D, RAW, LONG RAW, BLOB, CLOB, and NCLOB in Oracle.

In OCI programming LOB data can not simply be bound or fetched as part of a normal statement execution. Once a statement has been executed, extra OCI code is needed to perform the LOB operations (basically copying of the LOB data is done in a manual fashion).

Should a developer wish to add LOB support to the OCI Mapper, the methods of interest would be OD BIND TOWARDS 4D (for defining output parameters), OD BIND TOWARDS SQL (for defining input parameters), OD EXECUTE CURSOR (where the statement is actually executed), and OD Load rows cursor (where the data is fetched).

Important Method Notes for OCI Mapper 2004-3

This section covers important information about specific methods in the OCI Mapper 2004-3.

Method: **CheckError**

This method is the error handler used by the OCI Mapper to check 4D for OCI errors. It can also be installed as a custom error handler via the OD ON ERROR CALL command. This method was completely re-written for OCI Mapper 2004-3. The most important changes are:

- An error message is generated for all error cases.
- An OCI error handle can contain multiple error records. CheckError will now iterate through the error records in the error handle.
- Code was added to facilitate the use of OD Last Error.
- Removed all 4D for OCI calls (placed in oci_tool_GetErrorInfo for more general use)

If the developer is unfamiliar with OCI error handling it is highly recommended to take a good look at CheckError.

Method: **OD BIND TOWARDS 4D**

A note to 4D Oracle developers: the word “bind” here is a misnomer. In OCI programming output parameters are “defined”, not bound. Thus, a more correct name for this command might be “OD DEFINE OUTPUT PARAMETERS”.

Method: **OD Clone 4D Table**

This method is used to creates a copy of a 4D Table in Oracle. There are some important considerations however:

- Subtables are not supported.
- While the option exists in the UI to copy BLOBs and Pictures these types are not currently supported by the Mapper (see notes for LOBs).

Method: **OD Cursor state**

This method was unreliable in the previous version of the OCI Mapper because areas of the OCI Mapper code were not correctly updating the cursor state. This has been fixed.

Method: **OD Execute object**

This method does not support the execution of objects located in Oracle packages. Extra OCI code is needed to iterate through the package and locate the desired object.

Method: **OD GET SERVER LIST**

This method is used to get entries from the "tnsnames.ora" file. It uses the 4D for OCI method OCIGetTnsnamesPath to get the location of the Oracle Home directory. Unfortunately OCIGetTnsnamesPath does not currently support Oracle 10g. The location of the ORACLE_HOME registry key was changed in Oracle 10g so OCIGetTnsnamesPath does not look in the correct location to find the Oracle Home directory.

The developer can workaround this issue by either locating the Oracle Home in some other fashion or creating the registry key needed by OCIGetTnsnamesPath. The needed key is:

HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ORACLE_HOME

Method: **OD Last Error**

This method did not function at all in the previous version of the OCI Mapper. This has been fixed. See the header comments for CheckError for more information.

Note: "error" in this case means that the return value of a 4D for OCI call was OCI_ERROR. OD Last Error will not report things like OCI_NO_DATA or OCI_SUCCESS_WITH_INFO as these are not technically *errors*.

Note to 4D Oracle programmers: The only parameter supported from 4D Oracle is "message". The other parameters that existed in 4D Oracle do not exist in 4D for OCI. Also, the error codes returned will not be the same as in 4D Oracle. You will need to refer to your Oracle documentation for the possible error codes. E.g. a return of 936 means the error was ORA-00936.

Method: **OD Login**

Note to 4D Oracle programmers: the "mode" parameter from 4D Oracle no longer exists.

Method: **OD ON ERROR CALL**

Note to 4D Oracle programmers: the parameters passed to your custom error handler will be completely different than in 4D Oracle. Several of the

parameters that existed in 4D for Oracle simply do not exist in 4D for OCI. Here is the complete list of the parameters that existed in 4D for Oracle:

$1	Longint	Identifier of the object responsible for the error
$2	Longint	Code of the error that triggered the error call
$3	Text	Description of the error
$4	Longint	Error origin(100;200;300)
$5	Longint	Position of the error in the command, the number of the first character of the error from the beginning of the command text.
$6	Text	Name of the command responsible for the error

The parameters $1, $5, and $6 do not exist in 4D for OCI.

Your error handling method must be designed to accept two parameters as follows:

C_LONGINT($1)	Return code from the function that reported the error (e.g. OCI_ERROR, OCI_INVALID_HANDLE, etc.)
C_LONGINT($2)	OCI error handle (the error handle contains error "records" that have the actual error data in them).

For more details on what to do with the parameters see the source code for the CheckError method.

Method: **OD SET OPTIONS**

Note to 4D Oracle Developers: there are several options from 4D Oracle that are no longer available in the OCI Mapper:

Option	**Number**	**Description**
kNoWait	4	For contexts only. Specifies that an error must be returned if a lock is found upon opening a context for which the kWithLock option has been selected. The query is then of the SELECT...FOR UPDATE OF...NOWAIT type. In the opposite case, by default, the query will wait for the records to be unlocked.
kWithLock	8	For contexts only. Specifies that the context must lock the result rows. The query will be of the SELECT… FOR UPDATE OF… type if at least one bind has the kForUpdate option.
kDistinctLines	16	For contexts only. Specifies that the context must be activated by adding the DISTINCT keyword to the SELECT clause of the context (which removes duplicates in the result rows). The context is then automatically put in read-only mode.
kDeferred	64	Allows the program to use the OCI's deferred mode while sending requests to an Oracle7 server.

Also the options set with OD SET OPTIONS are set globally in the Mapper and do not apply to specific objects as in 4D Oracle.

OCI Mapper 2004-3 Command Reference

Note that only the high-level commands of the OCI Mapper are listed here. For information on the low-level, support commands in the OCI Mapper refer to the source code of the component itself.

All of the commands in the component contain header comments that describe the purpose of the command as well as list input/output parameters as appropriate.

OD BIND TOWARDS 4D
The OD BIND TOWARDS 4D method allows you to define output parameters.

OD BIND TOWARDS SQL
The OD BIND TOWARDS SQL method allows you to bind input parameters.

OD Clone 4D Table
The OD Clone 4D Table function creates an Oracle table with column definitions equivalent to a 4th Dimension table.

OD COMMIT
The OD COMMIT method calls OCITransCommit for a given connection.

OD Create cursor
The OD Create cursor function creates a cursor for the specified connection.

OD Cursor state
The OD Cursor state function informs the user of the state of the specified cursor.

OD DROP CURSOR
The OD DROP CURSOR method frees the memory used by a cursor that was previously created with OD Create cursor.

OD EXECUTE CURSOR
The OD EXECUTE CURSOR method executes the SQL statement associated with a cursor on the Oracle server.

OD Execute object
The command OD Execute object allows you to execute a stored procedure or stored function that is not located in an Oracle package.

OD Execute SQL
The OD Execute SQL function enables you to send a SQL query and store results in 4th Dimension fields, variables, or arrays, by specifying a login ID.

OD GET COLUMN ATTRIBUTES
The OD GET COLUMN ATTRIBUTES method allows you to retrieve the types and sizes of the result columns of a SQL query previously associated with a cursor. This method can be executed after the SQL statement has been set into cursor using OD Set SQL in cursor.

OD Get column title
The OD Get column title function allows you to retrieve the title of a result column following the execution of OD Set SQL in cursor.

OD Get NB Mode
The method OD Get NB Mode allows you to know the current status of the non-blocking mode.

OD Get options
The method OD Get options returns a longint representing the global options for the OCI Mapper.

OD GET SERVER LIST
The OD GET SERVER LIST method reads the TNSNAME.ORA file and returns in the array the list of server definition.

OD Last Error
The OD Last error function returns the number of the last 4D for OCI error. The error numbers will be listed in the Oracle documentation for the target server.

OD Load rows cursor
The OD Load rows cursor function takes advantage of array processing to load rows that have resulted from the execution of a SQL query using OD EXECUTE CURSOR.

OD Login
The OD Login function logs into an Oracle server using the login parameters you specify and returns a connection identifier.

OD Login dialog
The OD Login dialog function displays the Connect to an Oracle Server dialog box and allows the user to choose a server and to specify a login name and password.

OD Login state
The OD Login state function returns an integer indicating whether or not the specified connection ID refers to an open connection.

OD LOGOUT
The OD LOGOUT method ends the specified connection.

OD Number of columns
The OD Number of columns function returns the number of result columns in a query. This function can be used at any time after the OD Set SQL in cursor method has successfully executed.

OD Number rows processed
The OD Number rows processed function returns either the number of rows loaded by OD Load rows cursor since the execution of the cursor in the case of a query of the SELECT kind, or the number of inserted, modified, or deleted rows in the case of a query of the INSERT, UPDATE, or DELETE kind. Use this function after the query executes successfully.

OD ON ERROR CALL
The OD ON ERROR CALL method installs an error handling method that will be executed each time an error occurs. This allows you to control possible execution errors and override the default error handling.

OD ROLLBACK
The OD ROLLBACK method cancels an Oracle transaction.

OD SET NB MODE
The method OD SET NB MODE allows you to set the non-blocking mode.

OD SET OPTIONS
The OD SET OPTIONS method allows you to specify global options for the behavior of the OCI Mapper.

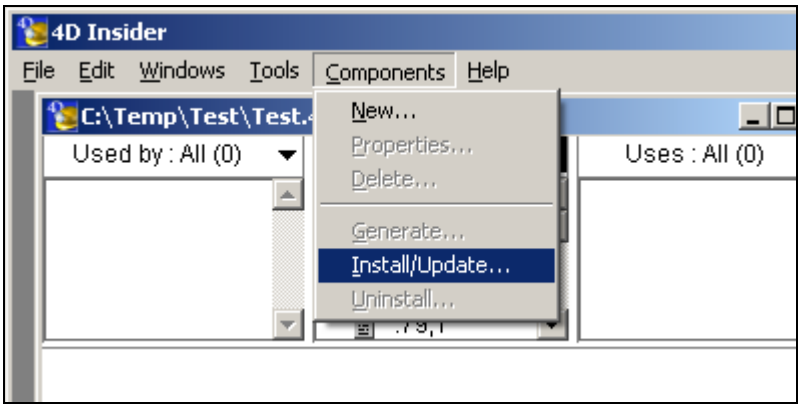
OD Set SQL in cursor
The OD Set SQL in cursor function allows you to associate a SQL statement with a cursor. The method sends the statement to the Oracle server. The statement is executed when you call OD EXECUTE CURSOR.

Installation Procedures for OCI Mapper 2004-3

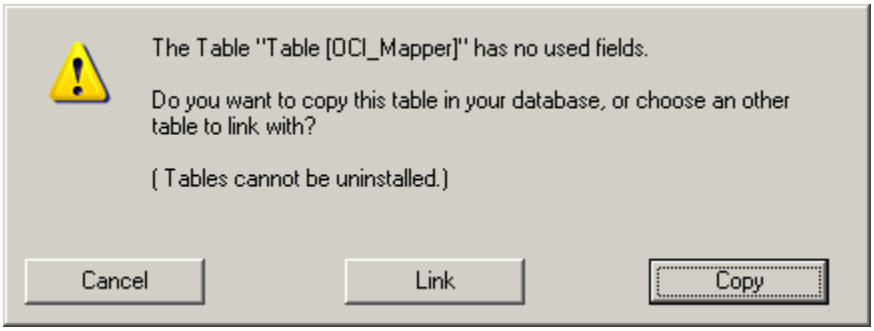
Note that 4D 2004 and 4D for OCI 2004 are required for the OCI Mapper 2004-3.

Installing the OCI Mapper into a new database

1. Open the database with 4D Insider.
2. Open the **Components** menu and select **Install/Update...**:



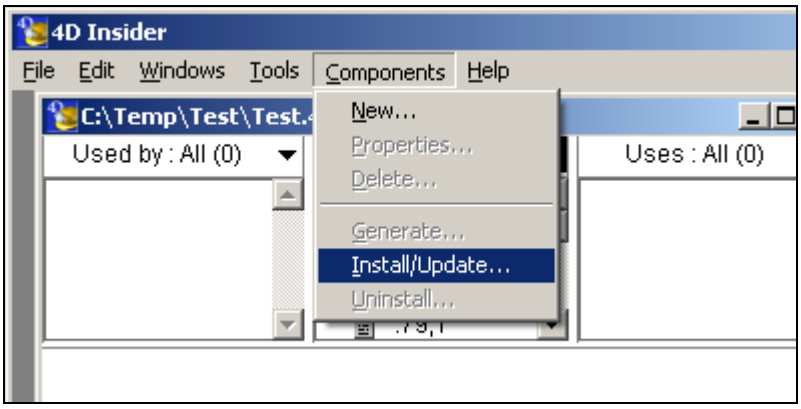
3. Browse for the "OCI Mapper 2004-3.4CP" file and click the **Open** button.
4. 4D Insider installs the OCI Mapper component. Be sure to copy the "OCI_Mapper" table:



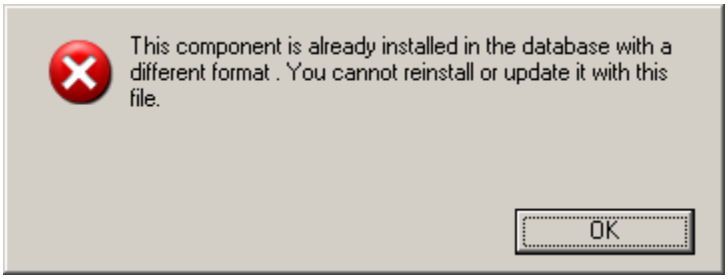
5. Quit 4D Insider.
6. Open the database with 4D.
7. Insert the method **OCI_TOOL_INITVAROCI** in the **On Startup** database method.
8. Finally be sure that the 4D for OCI plug-in is installed in the database.

Upgrading a previous version the OCI Mapper

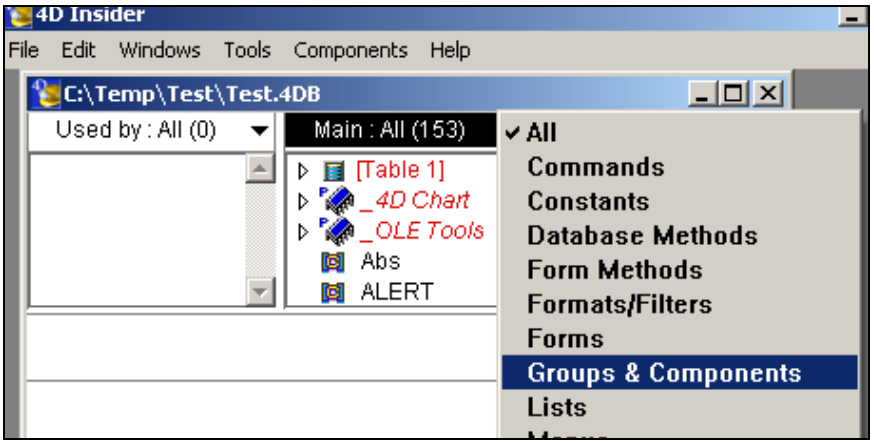
1. Open the database with 4D Insider.
2. Open the **Components** menu and select **Install/Update...**



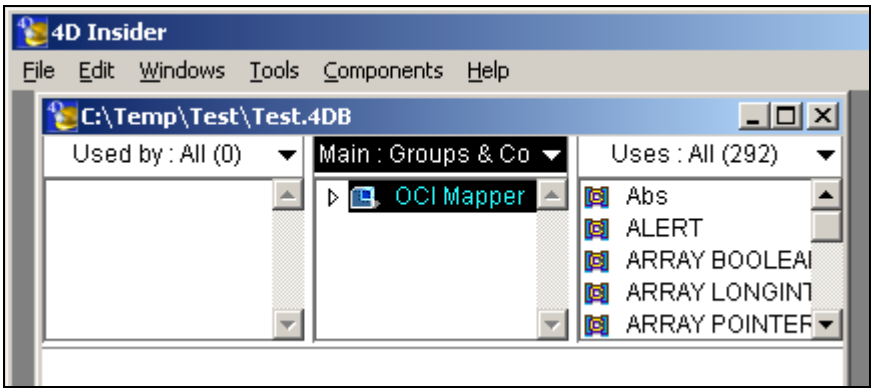
3. Browse for the "OCI Mapper 2004-3.4CP" file and click the **Open** button.
4. 4D Insider installs the OCI Mapper component. If you see this error move on to step 5:



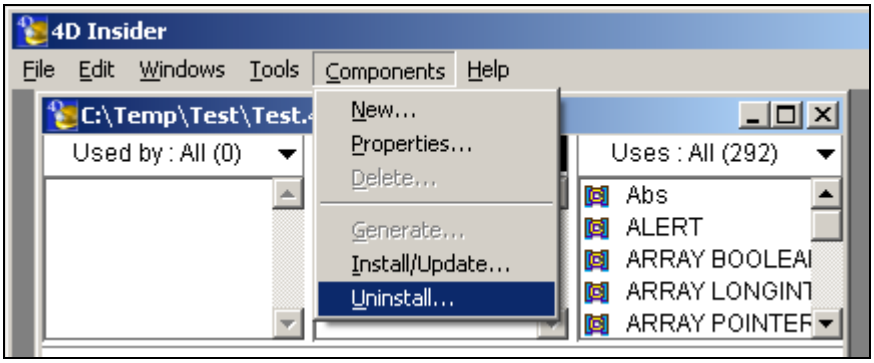
5. First uninstall the previous version of the OCI Mapper. Open the **Main** pop-up menu and select **Groups & Components**:



6. Select the **OCI Mapper** component:



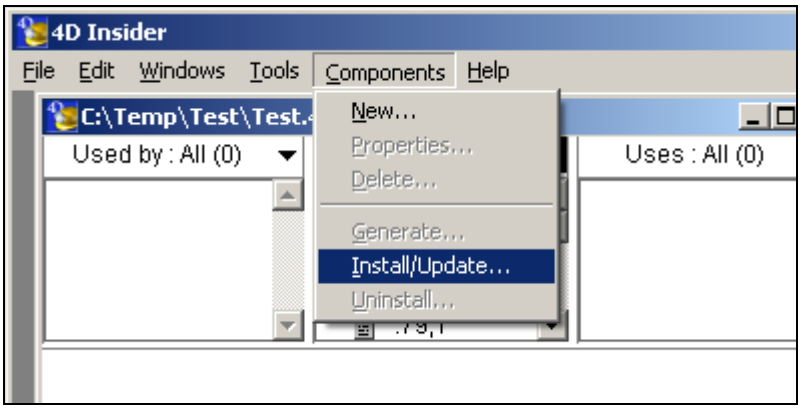
7. Open the **Components** menu and select **Uninstall...**:



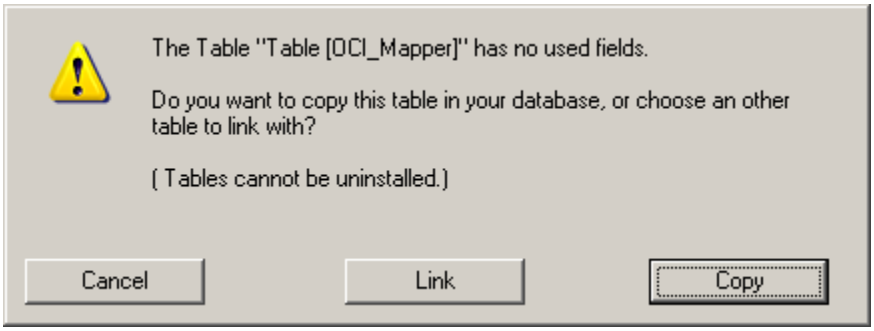
8. Click the **OK** button to uninstall the OCI Mapper.
9. Repeat steps 1 through 4 to install the new version of the OCI Mapper.

Upgrading a 4D Oracle database with the OCI Mapper

1. Open the database with 4D Insider.
2. Open the **Components** menu and select **Install/Update...**



3. Browse for the "OCI Mapper 2004-3.4CP" file and click the **Open** button.
4. 4D Insider installs the OCI Mapper component. Be sure to copy the "OCI_Mapper" table:



5. Quit 4D Insider.
6. Open the database with 4D.
7. Re-tokenize all methods that use 4D Oracle commands.
8. Quit 4D.
9. Remove the 4D Oracle plug-in in the Win4DX and/or Mac4DX folders.
10. Install the 4D for OCI plug-in.
11. Finally be sure to call **OCI_TOOL_INITVAROCI** at the startup of your database and **oci_tool_Initprocvar** at the start of each new process that will use the OCI Mapper.

Useful Resources

Useful 4D Resources

4D Oracle documentation:
ftp://ftp.4d.com/ACI_PRODUCT_REFERENCE_LIBRARY/4D_PRODUCT_DOCUMENTATION/PDF_Docs_by_4D_Product_A-Z/4D_Oracle/

4D for OCI Documentation:
http://www.4d.com/products/downloads_4d.html
(Included with the 4D 2004 All-in-One Installer package)

Useful Oracle Resources

Oracle Call Interface Programmer's Guide, 10*g* Release 2 (10.2):
http://download-west.oracle.com/docs/cd/B19306_01/appdev.102/b14250/toc.htm

Oracle Call Interface Programmer's Guide, Release 8.1.6
http://download-west.oracle.com/docs/cd/A87862_01/NT817CLI/index.htm

Conclusion

The OCI Mapper provides a way to easily migrate your database from using 4D Oracle to 4D for OCI. This can help minimize the time and cost in of 4D for OCI development. Additionally the OCI Mapper provides a good example of how to build a framework for 4D for OCI, which can ease the development phase and increase productivity.