Oracle® Call Interface

Getting Started

Release 9.2 for Windows

March 2002

Part No. A95497-01

Oracle Call Interface Getting Started, Release 9.2 for Windows

Part No. A95497-01

Contributors: Eric Belden, Janis Greenberg, Herbert Kelly, Helen Slattery, Ravi Thammaiah, and Bob Thome.

Contents

Send Us Your Comments

Oracle Call Interface Getting Started, Release 9.2 for Windows

Part No. A95497-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: ntdoc_us@oracle.com
- FAX: (650) 506-7365 Attn: Oracle Database for Windows Documentation
- Postal service:
  Oracle Corporation
  Oracle Database for Windows Documentation Manager
  500 Oracle Parkway, Mailstop 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This guide describes only the features of Oracle Call Interface (OCI) that apply to the Windows NT, Windows 2000, Windows XP, and Windows 98 operating systems. Information on Oracle9*i* Personal Edition software on Windows 98 is not covered in this guide.

This preface contains these topics:

- Audience
- Organization
- Related Documentation
- Conventions
- Documentation Accessibility

Audience

This guide is intended for developers who create applications written in C that interact with one or more Oracle Servers.

To use this document, you need to know:

- How to compile and link a C program.
- Your Microsoft Windows operating system.

Organization

This document contains:

Chapter 1, "Introduction to Oracle Call Interface"

Provides introductory information to help you get started with the OCI.

Chapter 2, "Building OCI Applications"

Provides an overview of how to build Oracle database applications using OCI.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Database Installation Guide for Windows*
- *Oracle9i Database Release Notes for Windows*
- *Oracle9i Database Administrator's Guide for Windows*
- *Oracle Enterprise Manager Administrator's Guide*
- *Oracle9i Net Services Administrator's Guide*
- *Oracle9i Real Application Clusters Concepts*
- *Oracle9i Database New Features*
- *Oracle9i Database Concepts*
- *Oracle9i Database Reference*
- *Oracle9i Database Error Messages*
- *Oracle Call Interface Programmer's Guide*

In North America, printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/admin/account/membership.html`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/docs/index.htm`

To access the database documentation search engine directly, please visit

`http://tahiti.oracle.com`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
UPPERCASE monospace (fixed-width font)	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a NUMBER column. You can back up the database by using the BACKUP command. Query the TABLE_NAME column in the USER_TABLES data dictionary view. Use the DBMS_STATS.GENERATE_STATS procedure.

Convention	Meaning	Example
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter sqlplus to open SQL*Plus. The password is specified in the orapwd file. Back up the datafiles and control files in the /disk1/oracle/dbs directory. The department_id, department_name, and location_id columns are in the hr.departments table. Set the QUERY_REWRITE_ENABLED initialization parameter to true. Connect as oe user. The JRepUtil class implements these methods.
lowercase monospace (fixed-width font) italic	Lowercase monospace italic font represents placeholders or variables.	You can specify the *parallel_clause*. Run U*old_release*.SQL where *old_release* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	DECIMAL (*digits* [, *precision*])
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	{ENABLE | DISABLE}
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]

Convention	Meaning	Example
. . .	Horizontal ellipsis points indicate either:	
	■ That we have omitted parts of the code that are not directly related to the example	CREATE TABLE ... AS *subquery*;
	■ That you can repeat a portion of the code	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	-

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	acctbal NUMBER(11,2);
		acct CONSTANT NUMBER(4) := 3;

Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	SELECT last_name, employee_id FROM employees;
		SELECT * FROM USER_TABLES;
		DROP TABLE hr.employees;

lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		SELECT last_name, employee_id FROM employees;
		sqlplus hr/hr
		CREATE USER mjones IDENTIFIED BY ty3MU9;

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose Start >	How to start a program. For example, to start Oracle Database Configuration Assistant, you must click the Start button on the taskbar and then choose Programs > Oracle - *HOME_NAME* > Database Administration > Database Configuration Assistant.	Choose Start > Programs > Oracle - *HOME_NAME* > Database Administration > Database Configuration Assistant
`C:\>`	Represents the Windows command prompt of the current hard disk drive. Your prompt reflects the subdirectory in which you are working. Referred to as the command prompt in this guide.	`C:\oracle\oradata>`

-	The backslash special character (\) is sometimes required as an escape character for the double quote (") special character at the Windows command prompt. Parentheses and the single quote special character (’) do not require an escape character. See your Windows operating system documentation for more information on escape and special characters.	`C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job=’SALESMAN’ and sal<1600\"`
		`C:\>imp SYSTEM/`*`password`*` FROMUSER=scott TABLES=(emp, dept)`

HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to 8.1, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default was: - C:\orant for Windows NT - C:\orawin95 for Windows 95 - C:\orawin98 for Windows 98 or whatever you called your Oracle home. In this Optimal Flexible Architecture (OFA)-compliant release, all subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is C:\oracle. If you install release 9.2 on a computer with no other Oracle software installed, the default setting for the first Oracle home directory is C:\oracle\ora90. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. See *Oracle9i Database Getting Started for Windows* for additional information on OFA compliances and for information on installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*\ORACLE_HOME\rdbms\admin directory.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

What’s New in Oracle Call Interface?

No new features have been added to this product for Oracle9*i* release 2 (9.2). Many new features were added for Oracle9*i* release 1 (9.0.1). If you are upgrading from a pre-9.0.1 release, read about the new features in Oracle9*i* release 1 (9.0.1) New Features in Oracle Call Interface release 9.0.1.

The following sections describe the new features in Oracle Call Interface:

- New Features in Oracle Call Interface for Release 9.0.1
- New Features in Oracle Call Interface for Release 8.1.5
- OCI Release 7.x Functions
- Oracle9i Release 2 (9.2) Deprecated Components

New Features in Oracle Call Interface for Release 9.0.1

This section contains these topics:

- **Borland Support**

  Oracle Corporation only ships an import library, `oci.lib`, for use with the Microsoft Compiler. Other compilers, for example, Borland, though likely compatible with the Oracle DLLs, are not tested and supported by Oracle for use with Oracle Call Interface.

- **Using Oracle9*i* on Windows 2000**

  There are some differences between using Oracle9*i* on Windows 2000 and Windows NT 4.0.

See Also: *Oracle9i Database Getting Started for Windows*

New Features in Oracle Call Interface for Release 8.1.5

OCI includes many new functions and performance enhancements that extend the capabilities of the OCI to handle objects in an Oracle8*i* database. To use object functionality, you must have installed Oracle8*i* Enterprise Edition.

For Windows platforms, OCI includes support for applications written with earlier releases (7.*x* / 8.*x*) of OCI. Oracle has now removed any version number from the library name `oci.lib`.

OCI Release 7.*x* Functions

OCI functions available in release 7.*x* are still available, but they are not able to take full advantage of new Oracle8*i* features. Oracle recommends applications start using the new calls to improve performance and provide increased functionality.

For Win32 applications running on Windows NT or Windows 95 or Windows 98, this means that these applications will need to migrate to the new release 8.*x* OCI calls in order to continue to be supported. In release 8.*x*, the library and DLL containing the OCI calls is named `oci.lib` and `oci.dll`. In release 7.*x*, they were named `ociw32.lib` and `ociw32.dll`. At some point in the future, `ociw32.lib` and `ociw32.dll` will no longer be supported or released, making migration to the new calls mandatory.

Oracle9*i* Release 2 (9.2) Deprecated Components

The Intype File Assistant (IFA) is deprecated in this release and it will not be supported in future releases.

1

Introduction to Oracle Call Interface

This chapter provides introductory information to help you get started with Oracle Call Interface (OCI) for Windows.

This chapter contains these topics:

- What is the Oracle Call Interface?
- What is Included in the OCI Package?
- Oracle Directory Structure
- Sample Programs

See Also: *Oracle Call Interface Programmer's Guide* for detailed information about OCI, including new features and function descriptions

What is the Oracle Call Interface?

The Oracle Call Interface (OCI) is an application programming interface (API) that allows applications written in C to interact with one or more Oracle Servers. OCI gives your programs the capability to perform the full range of database operations that are possible with Oracle9*i* database, including SQL statement processing and object manipulation.

What is Included in the OCI Package?

The Oracle Call Interface for Windows package includes:

- Oracle Call Interface
- Required Support Files (RSFs)
- Oracle Universal Installer
- Header files for compiling OCI applications
- Library files for linking OCI applications
- Sample programs for demonstrating how to build OCI applications

The OCI for Windows package includes the additional libraries required for linking your OCI programs on Windows NT, Windows 2000, and Windows 98.

Oracle Directory Structure

When you install the Oracle Call Interface for Windows, Oracle Universal Installer creates an *`ORACLE_BASE\ORACLE_HOME`* directory on the hard drive of your computer. The default Oracle home directory is `C:\oracle\ora92`.

The OCI files are located in the *`ORACLE_BASE\ORACLE_HOME`* directory, as are the library files needed to link and run OCI applications, and link with other Oracle for Windows NT products, such as Oracle Forms.

The *`ORACLE_BASE\ORACLE_HOME`* directory contains the following directories that are relevant to OCI:

Table 1–1 Oracle_Home Directories and contents

Directory Name	Contents
`\bin`	Executable and help files
`\oci`	Oracle Call Interface directory for Windows files

Table 1–1 Oracle_Home Directories and contents (Cont.)

Directory Name	Contents
\oci\include	Header files, such as oci.h and ociap.h
\oci\samples	Sample programs
\precomp\admin\ottcfg.cfg	Object Type Translator utility and default configuration file

Sample Programs

When OCI is installed, a set of sample programs and their corresponding project files are copied to the *ORACLE_BASE*\ORACLE_HOME\oci\samples subdirectory. Oracle Corporation recommends that you build and run these sample programs to verify that OCI has been successfully installed and to familiarize yourself with the steps involved in developing OCI applications.

To build a sample, run a batch file(make.bat)at the command prompt. For example, to build the cdemo1.c sample, enter the following command:

```
C:> make cdemo1
```

After you finish using these sample programs, you can delete them if you choose.

A sample OCI application specific to Windows platforms is included. cdemomt.c demonstrates OCI multithreading which is the thread safety feature of Oracle9*i* is also included on the Windows platforms. This sample program requires the emp table from the default database. The program spawns two simultaneous threads that attempt to insert different employee names with the same ID numbers. Thread synchronization is demonstrated.

ociucb.c should be compiled using ociucb.bat. This batch file creates a DLL and places it in the *ORACLE_BASE*\ORACLE_HOME\bin directory. To load user callback functions, set the environment/registry variable ORA_OCI_UCBPKG = OCIUCB.

> **See Also:** *Oracle Call Interface Programmer's Guide* for more information on multithreading

2

Building OCI Applications

This chapter provides an overview of how to build Oracle database applications using OCI.

This chapter contains these topics:

- Writing OCI Applications
- Compiling OCI Applications
- Linking OCI Applications
- The Oracle XA Library
- Using the Object Type Translator and the INTYPE File Assistant

> **See Also:** *Oracle Call Interface Programmer's Guide* for detailed information about writing OCI applications

Writing OCI Applications

The general goal of an OCI application is to connect to an Oracle Server, engage in some sort of data exchange, and perform necessary data processing. While some flexibility exists in the order in which specific tasks can be performed, every OCI application must accomplish particular steps.

The basic programming structure used by the OCI is as follows:

1. Initialize the OCI programming environment and processes.
2. Allocate necessary handles, and establish a server connection and a user session.
3. Issue SQL statements to the server, and perform necessary application data processing.
4. Free statements and handles not to be reused or reexecute prepared statements again, or prepare a new statement.
5. Terminate user session and server connection.

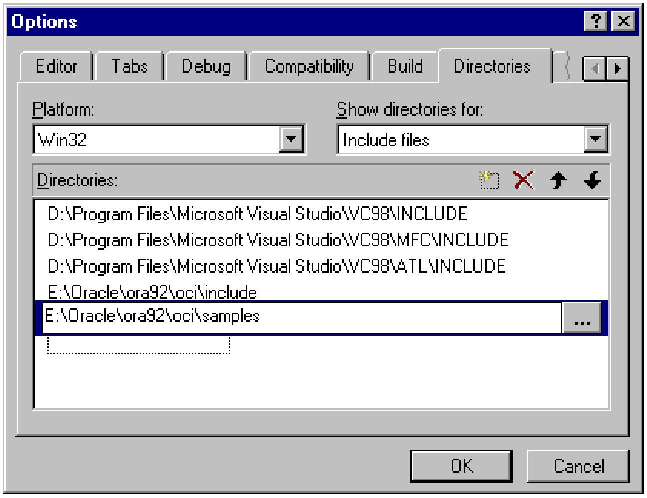
Note: The initialization of an OCI environment in Shared Data Mode that is discussed in the *Oracle Call Interface Programmer's Guide* is not supported on Windows.

Compiling OCI Applications

When you compile an OCI application, you must include the appropriate OCI header files. The header files are located in the `\`*`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\oci\include` directory.

For example, if you are using Microsoft Visual C++ 6.0, you would need to put in the appropriate path in the Directories page of the Options dialog in the Tools menu. See Figure 2–1, "Directories Tab of the Options Dialog".

Figure 2–1 Directories Tab of the Options Dialog



Platform: Win32

Show directories for: Include files

Directories:

```
D:\Program Files\Microsoft Visual Studio\VC98\INCLUDE
D:\Program Files\Microsoft Visual Studio\VC98\MFC\INCLUDE
D:\Program Files\Microsoft Visual Studio\VC98\ATL\INCLUDE
E:\Oracle\ora92\oci\include
E:\Oracle\ora92\oci\samples
```

> **See Also:** Your compiler's documentation for specific information about compiling your application and special compiler options

Linking OCI Applications

The OCI calls are implemented in dynamic link libraries (DLLs) that Oracle provides. The DLLs are located in the `*ORACLE_BASE*\*ORACLE_HOME*\bin` directory and are part of the Required Support Files (RSFs).

To use the Oracle DLLs to make OCI calls, you can either dynamically load the DLL and function entry points, or you can link your application with the import library `oci.lib`. Oracle Corporation only provides the `oci.lib` import library for use with the Microsoft Compiler. Other compilers, though likely compatible with the Oracle DLLs, are not tested and supported by Oracle for use with OCI.

When using `oci.lib` with the Microsoft Compiler, you do not have to indicate any special link options.

oci.lib

`oci.lib` is a single, programmatic interface to Oracle. Oracle has removed any version number from the library name.

Client DLL Loading When Using LoadLibrary()

The following directories are searched in this order by `LoadLibrary`:

- Directory from which the application is loaded
- Current directory
- Windows NT or Windows 2000:
  - 32-bit Windows system directory `(system32)`. Use the `GetWindowsDirectory` function to obtain the path of this directory.
  - 16-bit Windows directory `(system)`. There is no Win32 function that obtains the path of this directory, but it is searched.
- Windows 98:
  - Windows directory. Use the `GetWindowsDirectory` function to obtain the path of this directory.
- Directories that are listed in the PATH environment variable

Running OCI Applications

To run an OCI application, ensure that the entire corresponding set of Required Support Files (RSFs) is installed on the computer that is running your OCI application.

The Oracle XA Library

The XA Application Program Interface (API) is typically used to enable an Oracle9*i* database to interact with a transaction processing (TP) monitor, such as:

- BEA Tuxedo
- IBM Transarc Encina
- IBM CICS

You can also use TP monitor statements in your client programs. The use of the XA API is supported from OCI.

The Oracle XA Library is automatically installed as part of Oracle9*i* Enterprise Edition. The following components are created in your Oracle home directory:

Table 2–1 Oracle XA Library Components

Component	Location
oraxa9.lib	*ORACLE_BASE*\ORACLE_HOME\rdbms\xa
xa.h	*ORACLE_BASE*\ORACLE_HOME\rdbms\demo

Compiling and Linking an OCI Program with the Oracle XA Library

To compile and link an OCI program:

1. Compile `program.c` by using Microsoft Visual C++, making sure to include *ORACLE_BASE*\ORACLE_HOME\rdbms\xa in your path.
2. Link `program.obj` with the following libraries:

Library	Located in...
oraxa9.lib	ORACLE_BASE\ORACLE_HOME\rdbms\xa
oci.lib	ORACLE_BASE\ORACLE_HOME\oci\lib\msvc

3. Run `program.exe`.

Using XA Dynamic Registration

The Oracle9*i* database supports the use of XA dynamic registration. XA dynamic registration improves the performance of applications interfacing with XA-compliant TP monitors. For TP Monitors to use XA dynamic registration with an Oracle database on Windows NT, you must add either an environmental variable or a registry variable to the Windows NT computer on which your TP monitor is running. See either of the following sections for instructions:

- Adding an Environmental Variable for the Current Session
- Adding a Registry Variable for All Sessions

Adding an Environmental Variable for the Current Session

Adding an environmental variable at the command prompt affects only the current session.

To add an environmental variable:

From the computer where your TP monitor is installed, enter the following at the command prompt:

```
C:\> set ORA_XA_REG_DLL = vendor.dll
```

where *vendor*.dll is the TP monitor DLL provided by your vendor.

Adding a Registry Variable for All Sessions

Adding a registry variable affects all sessions on your Windows NT computer. This is useful for computers where only one TP monitor is running.

To add a registry variable:

1. Go to the computer where your TP monitor is installed.
2. On Windows NT or Windows 2000, enter the following at the command prompt:

   ```
   C:\> regedt32
   ```

   On Windows 98, enter:

   ```
   C:\> regedit
   ```

   The Registry Editor window appears.
3. Go to HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOME*ID*.
4. Choose the Add Value option in the Edit menu. The Add Value dialog box appears.
5. Enter ORA_XA_REG_DLL in the Value Name text box.
6. Select REG_EXPAND_SZ from the Data Type list box.
7. Choose OK. The String Editor dialog box appears.
8. Type *vendor*.dll in the String field, where *vendor*.dll is the TP monitor DLL provided by your vendor.

9. Choose OK. The Registry Editor adds the parameter.
10. Choose Exit from the Registry menu.

    The registry exits.

XA and TP Monitor Information

Refer to the following general information about XA and TP monitors:

- *Distributed TP: The XA Specification* (C193) published by the Open Group. The Open Group., 1010 El Camino Real, Suite 380, Menlo Park, CA 94025, U.S.A.
- See the Web site at:

  `http://www.opengroup.org/publications/catalog/tp.htm`
- Your specific TP monitor documentation

Oracle C++ Call Interface Methods Specific to Windows NT

The global methods for getting collections of `Refs` or setting collections of `Refs` from classes `Statement` and `ResultSet` have changed for Windows NT as follows:

- Use `getVectorOfRefs` in place of `getVector` on Windows NT
- Use `setVectorOfRefs` in place of `setVector` on Windows NT

The method names have been changed but the number of parameters and the types of the parameters remain the same as the original `getVector` and `setVector` methods for `Refs` on these classes.

- Applications on Windows NT platforms should be calling these new methods only for retrieving and inserting collections of `Refs`.
- Applications not running on Windows NT platforms have a choice of calling the currently existing `getVector` and `setVector` methods. However, Oracle Corporation recommends the use of the new methods for any vector operations with `Refs`.

ResultSet Class: Fetching collection of Refs

```
void getVectorOfRefs(ResultSet  *rs, unsigned int index,
 OCCI_STD_NAMESPACE::vector<Ref<T> > &vect);
```

This method fetches a column value specified by the column index that is a collection of `Refs` from a result set.

The parameters are:

- rs - ResultSet object
- index - the column index of a column which is a collection of Refs
- vect - the vector into which the Refs are fetched

Statement Class: Fetching collection of Refs

```
void getVectorOfRefs(Statement *stmt, unsigned int index,
 OCCI_STD_NAMESPACE::vector<Ref<T> > &vect);
```

This method fetches a column value specified by the column index that is a collection of Refs from a statement. This is used in case of OUT binds and data manipulation language (DML) returning clauses. The parameters are:

- stmt - statement object
- index - the column index of a column that is a collection of Refs
- vect - the vector into which the Refs are fetched

Statement Class: Inserting a collection of Refs

```
template  <class T>
void setVectorOfRefs(Statement *stmt, unsigned int paramIndex,
  const OCCI_STD_NAMESPACE::vector<Ref<T> > &vect,
  const OCCI_STD_NAMESPACE::string &sqltype);
```

This method inserts a collection of Refs into a column specified by the index. The parameters are:

- stmt - statement object
- paramIndex - the column index of a column that is a collection of Refs
- vect - the vector of Refs that are inserted into the column
- sqltype - the type name of the collection that was created in the database

The global methods for the fetching or inserting of collections of objects have been changed for Windows NT. The interface remains the same with respect to the method names and the number of parameters and the datatypes, but differs in the template parameter definition for Windows NT. Specifically, the template parameter for the template methods of getVector and setVector of objects (object pointers) on Windows NT have a T instead of a T * as shown in the following APIs.

Note that the usage of the methods does not differ across the platforms (users need not modify the call to these methods at all). On Windows NT, the template arguments passed as object pointers in the method call are specialized for the parameter T instead of T * on other platforms.

class ResultSet: fetching a collection of objects

```
#ifdef WIN32COMMON
   template <class T>
   void getVector( ResultSet *rs, unsigned int index,
     OCCI_STD_NAMESPACE::vector< T > &vect) ;
#else
  template <class T>
    void getVector( ResultSet *rs, unsigned int index,
      OCCI_STD_NAMESPACE::vector< T* > &vect) ;
#endif
```

This method fetches a collection of objects from a `ResultSet` for the column specified by the `index`.

The parameters are:

- `rs` - resultSet object
- `index` - column index
- `vect` - the vector into which the objects should be fetched

class Statement: fetching a collection of objects

```
#ifdef WIN32COMMON
   template <class T>
   void getVector( Statement *stmt, unsigned int index,
     OCCI_STD_NAMESPACE::vector< T > &vect) ;
  #else
   template <class T>
   void getVector( Statement *stmt, unsigned int index,
     OCCI_STD_NAMESPACE::vector< T* > &vect) ;
  #endif
```

This method fetches a collection of objects from a statement for the column specified by the `index`. This method is used in case of OUT binds and DML returning clauses. The parameters are:

- `stmt` - statement object
- `index` - column index
- `vect` - the vector into which the objects should be fetched

class Statement: inserting a vector of objects

```
#ifdef WIN32COMMON
  template <class T>
  void setVector( Statement *stmt, unsigned int paramIndex,
    const OCCI_STD_NAMESPACE::vector< T > &vect,
    const  OCCI_STD_NAMESPACE::string &sqltype) ;
#else
  template <class T>
  void setVector( Statement *stmt, unsigned int paramIndex,
    const OCCI_STD_NAMESPACE::vector<T* > &vect,
    const OCCI_STD_NAMESPACE::string &sqltype) ;
#endif
```

This method inserts a collection of objects into a statement for the column specified by the `index`. The parameters are:

- `stmt` - statement object
- `paramIndex` - column index
- `vect` - the vector into which the objects should be fetched
- `sqltype` - the type name of the collection that was created in the database

> **See Also:** *Oracle9i Application Developer's Guide* - *Fundamentals* for more information about the Oracle XA Library and using XA dynamic registration

Using the Object Type Translator and the INTYPE File Assistant

The Object Type Translator (OTT) is used to create C-struct representations of Abstract Data Types that have been created and stored in an Oracle9*i* database.

To take advantage of objects run OTT against the database, and a header file is generated that includes the C structs. For example, if a PERSON type has been created in the database, OTT can generate a C struct with elements corresponding to the attributes of PERSON. In addition, a null indicator struct is created that represents null information for an instance of the C struct.

The INTYPE file tells the OTT which object types should be translated. This file also controls the naming of the generated structs. The INTYPE File Assistant is a wizard that helps developers to create the INTYPE file.

Note that the `CASE` specification inside the `INTYPE` files, such as `CASE=LOWER`, applies only to C identifiers that are not specifically listed, either through a `TYPE` or `TRANSLATE` statement in the INTYPE file. It is important to provide the type name with the appropriate cases, such as TYPE Person and Type PeRsOn, in the INTYPE file.

The INTYPE File Assistant generates type names in the INTYPE file with the same case as in the database. By default, all of the types in the database are created in upper case.

In order to preserve the case, use double quotes when creating types in the database. For example:

```
CREATE TYPE "PeRsOn" AS OBJECT...
```

Object type dependencies are not checked by the Oracle INTYPE File Assistant. When adding an object type for inclusion in the INTYPE file, the INTYPE File Assistant does not add other object types with dependency relationships.

The INTYPE File Assistant requires explicit translations for object types or attributes whose names contain non-ASCII characters. These object types or attributes are indicated by the predefined tag Identifier in the fields where the translations would be entered. Users are required to override this tag with the C identifier translation for the corresponding object type or attribute. The INTYPE File Assistant does not create the INTYPE file until all required translations have been entered.

OTT on Windows NT can be invoked from the command line. Additionally, a configuration file may be named on the command line. For Windows NT, the configuration file is `ottcfg.cfg`, located in *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\precomp\admin`.

The Intype File Assistant (IFA) is deprecated in this release and it will no longer be supported in release 10i.

> **See Also:** *Oracle Call Interface Programmer's Guide* for more information about OTT and INTYPE files

Index