

Oracle[®] COM Automation

Developer's Guide

Release 8.1.6 for Windows NT

January 2000

Part No. A73027-01

Part No. A73027-01

Copyright © 1999, 2000 Oracle Corporation. All rights reserved.

Contributors: Eric Belden, Kin Lau, Kian Fai Leong, Barmak Meftah, Steve Norall, Jeff Stein

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Net8, Oracle8, Oracle8i, PL/SQL, Trusted Oracle, and SQL*Plus are trademarks or registered trademarks of Oracle Corporation. All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

Contents

Contact Us!	vii
How to Contact Oracle Technical Publications	viii
How to Contact Oracle Support Services	ix
Resources for Oracle Partners and Developers	xiii
Preface	xvii
Prerequisites	xviii
Intended Audience	xviii
How This Guide Is Organized	xviii
Conventions	xix
Documentation Library	xxi
Related Documents	xxii
1 Introducing Oracle COM Automation	
Introducing Oracle COM Automation Feature	1-2
Updates for Release 8.1.6	1-2
What Is Oracle COM Automation Feature?	1-2
Benefits of Oracle COM Automation Feature	1-3
Oracle COM Automation Feature Architecture	1-4
Invoking OLE Automation External Procedure APIs	1-5
Architectural Impact on Availability and Performance Issues	1-6
Oracle Database Availability	1-6
Oracle COM Automation Feature Performance	1-7

2	Installing Oracle COM Automation	
	Oracle COM Automation Feature Components	2-2
	System Requirements	2-2
3	Post-Installation Configuration Tasks	
	Optimal Flexible Architecture for Oracle8i	3-2
	Configuring Oracle COM Automation Feature	3-2
	Configuring the Listener	3-3
	ORA-28575: Error Message	3-4
	Configuring DCOM	3-5
4	Oracle COM Automation Core Functionality	
	Oracle COM Automation Feature Core Functionality	4-2
	Developing Solutions Using Oracle COM Automation	4-3
	Information Required for COM Objects	4-3
	OLE/COM Object Viewer	4-4
	Datatype Conversion	4-5
	HRESULT Return Codes	4-6
	PL/SQL Application Programming Interfaces	4-6
	CreateObject	4-6
	DestroyObject	4-8
	GetLastError	4-8
	GetProperty	4-10
	SetProperty	4-11
	InitArg	4-12
	SetArg	4-13
	SetPtrArg	4-14
	Invoke	4-15
	Oracle COM Automation Errors	4-16
	OLE Automation Errors	4-18
5	Using Oracle COM Automation Demos	
	Oracle COM Automation Feature Demos Overview	5-2
	Microsoft Word Demo	5-2

Installing the Microsoft Word Demo	5-2
Using the Microsoft Word Demo	5-3
Core Functionality	5-4
Microsoft Excel Demo	5-7
Installing the Microsoft Excel Demo	5-7
Using the Microsoft Excel Demo	5-7
Core Functionality	5-8
Microsoft PowerPoint Demo	5-11
Installing the Microsoft PowerPoint Demo	5-11
Using the Microsoft PowerPoint Demo	5-12
Core Functionality	5-12
MAPI Demo	5-15
Installing the MAPI Demo	5-15
Using the MAPI Demo	5-16
Core Functionality	5-16

Glossary

Index

Contact Us!

Oracle COM Automation Developer's Guide, Release 8.1.6 for Windows NT

Part No. A73027-01

This document describes how to contact Oracle Corporation if you have issues with the documentation or software.

Read the section...	If you...
How to Contact Oracle Technical Publications on page viii	Have issues with Documentation
How to Contact Oracle Support Services on page ix	Have issues with Software
Resources for Oracle Partners and Developers on page xiii	Want to join an Oracle partner or application developer program

How to Contact Oracle Technical Publications

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. 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 about this guide?
- Do you have suggestions for improvement? Please indicate the chapter, section, and page number (if available).

You can send comments regarding documentation in the following ways:

- Electronic mail - ntdoc@us.oracle.com
- FAX - (650) 506-7370 Attn: Oracle Windows Platforms Server Documentation
- Postal service:
Oracle Corporation
Windows Platforms Server Documentation Manager
500 Oracle Parkway, MS 1OP8,
Redwood Shores, CA 94065
USA

If you would like a reply, please provide your name, address, and telephone number.

How to Contact Oracle Support Services

Please copy this form and distribute within your organization as necessary.

Oracle Support Services can be reached at the following telephone numbers and Web sites. The hours of business are detailed in your support contract and the *Oracle Customer Support Guide* in your kit.

Oracle Support Services In...	Call...
United States of America	+ (650) 506-1500 for customers with support contracts. + (650) 506-5577 to obtain a support contract.
Europe	+44 1344 860 160 or the local support center in your country.
All other locations	The telephone number for your country listed at the following Web site: http://www.oracle.com/support/contact_us/sup_hot_phone.html Oracle Support Services telephone numbers are also listed in the <i>Oracle Customer Support Guide</i> in your kit.

Please complete the following checklist before you call. If you have this information ready, your call can be processed much quicker.

- Your CPU Support Identification Number (CSI Number) if applicable.

- The hardware name on which your application is running.

-
- ❑ The operating system name and release number on which your application is running.

- To verify the operating system version on Windows NT, enter the following at the MS-DOS command prompt:

```
C:\> winmsd
```

The *Windows NT Diagnostics* dialog box displays the operating system and Service Pack version.

-
-
-
- ❑ The release numbers of the Oracle Server and associated products involved in the current problem. For example, Oracle8i Enterprise Edition release 8.1.6.0.0 and Oracle Enterprise Manager release 2.1.0.0.0.

- To verify the release number of the Oracle Server, connect to the database using a tool such as SQL*Plus. The release number is displayed. For example:

```
Connected to:
Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.6.0.0 - Production
```

-
-
-
- ❑ The third-party software version you are using.

- To verify an application version, from the application's Help menu, select About...
-
-
-

-
- ❑ The exact error codes and messages. Please write these down as they occur. They are critical in helping Oracle Support Services to quickly resolve your problem. Note whether there were no errors reported.

- ❑ A description of the issue, including:

- **What happened?** For example, the command used and its result.

- **When did it happen?** For example, during peak system load, or after a certain command, or after an operating system upgrade. In addition, what was happening when the problem occurred?

- **Where did it happen?** For example, on a particular system, or within a certain procedure or table.

-
- **What is the extent of the problem?** For example, production system unavailable, or moderate impact but increasing with time, or minimal impact and stable.
 - Did the problem affect one user, several users, or all users?
 - Has anything changed? For example, if this is an operation that used to work and now fails, what is different? Can you undo any recent changes, to verify whether they are relevant to the issue?

- **Can the problem be reproduced?** This is a critical question for support analysts. For example, did the problem recur on the same system, under the same circumstances? Can the problem be reproduced on another system? Additionally:
 - Does installing a software component fail on all client machines, or just one?
 - Do all clients fail to connect to the server, or just one?
 - If you are able to restart the server or database, does restarting the database or rebooting the server or client machine (if applicable) make a difference?

- Keep copies of the Oracle alert log, any trace files, core dumps, and redo log files recorded at or near the time of the incident. Oracle Support Services may need these to further investigate your problem.

To help analyze problems:

- Archive or delete old alert logs. When the database is started without an alert log, a new one is created. In some cases, if you force the problem to recur with a new alert log, the timestamps for the recorded events may indicate which events are relevant.

- Archive or delete old trace files. To check whether the file was modified, right-click and select Properties. The *Properties* dialog box displays the modification date.
- Check the operating system error logs, especially the System log and Application log. These files are relevant to the Oracle Server. To view these files, from the Start menu, choose Programs > Administrative Tools > Event Viewer, and choose System or Application from the Log main menu.

Resources for Oracle Partners and Developers

This section provides information on partner programs and resources for Oracle database administrators and application developers.

Information Source	Description
Oracle Corporation Home Page http://www.oracle.com	This Web site is the starting point for general information on Oracle Corporation.
Alliance Online http://alliance.oracle.com	Oracle provides leading-edge technology, education, and technical support that enables you to effectively integrate Oracle into your business. By joining the Oracle Partner Program, you demonstrate to customers that you are committed to delivering innovative Oracle-based solutions and services. The greater your commitment to Oracle, the more we can help you grow your business. It's that simple. The value you derive is associated directly with your level of commitment.
Oracle Education http://education.oracle.com/	Customers come to Oracle Education with a variety of needs. You may require a complete curriculum based on your job role to enable you to implement new technology. Or you may seek an understanding of technology related to your key area of responsibility to help you meet technical challenges. You may be looking for self-paced training that can be used as an ongoing resource for reference and hands-on practice. Or, you may be interested in an overview of a new product upgrade. Whatever your training need, Oracle Education has the solution.

Information Source	Description
Oracle Technology Network http://technet.oracle.com/	The Oracle Technology Network is your definitive source for Oracle technical information for developing for the Internet platform. You will be part of an online community with access to free software, Oracle Technology Network-sponsored Internet developer conferences, and discussion groups on up-to-date Oracle technology. Membership is free.
Oracle Store http://oraclestore.oracle.com/	This is Oracle's online shopping center. Come to this site to find special deals on Oracle software, documentation, publications, computer-based training products, and much more.
Oracle Support Services' Support Web Center http://www.oracle.com/support/	Oracle Support Services offers a range of programs so you can select the support services you need and access them in the way you prefer: by telephone, electronically, or face to face. These award-winning programs help you maintain your investment in Oracle technology and expertise. Here are some of the resources available in the Support Web Center:
OracleMetaLink http://www.oracle.com/support/elec_sup/index.html	OracleMetaLink is Oracle Support Services' premier Web support service. It is available to Oracle <i>metals</i> customers (Gold, Silver, Bronze), 24 hours a day, seven days a week.
OracleLifecycle http://www.oracle.com/support/sup_serv/lifecycle/index.html	OracleLifecycle is designed to deliver customized, industry-focused, full life-cycle support solutions that enable industry leaders to use Oracle technology to make smart business decisions, achieve operational excellence, and succeed in their markets.
ExpertONLINE http://www.oracle.com/support/sup_serv/online/index.html	Oracle Support Services has launched a new line of services called ExpertONLINE . These services provide online database administration for companies looking to supplement their existing DBA staff or fill a DBA role. Services range from ExpertDETECT , a monitoring, diagnostic, and recommendation service, to ExpertDBA , a full online database administration service.
Virtual Support Analyst (VSA) http://www.oracle.com/support/sup_serv/vsa_start.html	VSA is Oracle's Internet e-mail service; it is available to U.S. customers with an Oracle <i>metals</i> support agreement. With VSA , you can initiate a request for assistance through e-mail, bypassing the queues you may encounter when using telephone support. VSA also enables you to access Oracle's bug database.

Information Source	Description
<p>Customer Service</p> <p>http://www.oracle.com/support/cus_serv/index.html</p>	<p>This site provides resources to make your interactions with Oracle as easy as possible. Among the things you can do are:</p> <ul style="list-style-type: none"> ■ Learn what is a CPU Support Identification (CSI) number ■ Update your technical contact information ■ Find out whom to contact for invoice and collection issues ■ Request product update shipments ■ Access a glossary of Oracle Support Services terms
<p>U.S. Customer Visit Program</p> <p>http://www.oracle.com/support/cus_serv/cus_visit.html</p>	<p>This U.S.-based program has been established to help our customers understand and obtain maximum benefit from the support services they have purchased.</p> <p>The visit typically offers a customized orientation presentation, a comprehensive overview and demonstration of Oracle's electronic services, and helpful tips on working more effectively with Oracle Support Services.</p>
<p>Support Web Center Library</p> <p>http://www.oracle.com/support/library/index.html</p>	<p>This site contains articles, guides, and other documentation to help you leverage the wealth of knowledge and reference material that Oracle Support Services produces.</p>



Preface

This document is your primary source of introductory, installation, post-installation configuration, and usage information for Oracle COM Automation feature. Specific topics discussed in this preface are:

- [Prerequisites](#)
- [Intended Audience](#)
- [How This Guide Is Organized](#)
- [Conventions](#)
- [Documentation Library](#)
- [Related Documents](#)

Prerequisites

This document assumes that you are familiar with the following technologies:

- Component Object Model (COM)
- Object Linking and Embedding (OLE) Automation
- Structured Query Language (SQL)
- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- PL/SQL
- Oracle object-relational database management (ORDBMS) concepts
- Windows NT

Intended Audience

The Oracle COM Automation feature Software Development Kit (SDK) release is intended for developers who develop solutions that use COM.

How This Guide Is Organized

This guide is organized as follows:

Chapter 1, "Introducing Oracle COM Automation"

Provides an overview of Oracle COM Automation feature and Oracle Server architecture. Read this chapter *before* installing or using Oracle COM Automation feature.

Chapter 2, "Installing Oracle COM Automation"

Describes how to install Oracle COM Automation feature. This chapter also lists the contents of the Oracle COM Automation feature SDK and describes the system requirements.

Chapter 3, "Post-Installation Configuration Tasks"

Describes the configuration tasks you *must* perform before using Oracle COM Automation feature.

Chapter 4, "Oracle COM Automation Core Functionality"

Describes the core functionality of Oracle COM Automation feature and the PL/SQL APIs for manipulating COM objects using the OLE Automation interface.

Chapter 5, "Using Oracle COM Automation Demos"

Describes how to use Oracle COM Automation feature demos.

Glossary

Describes terms used in this document.

Conventions

The following conventions are used in this guide.

Convention	Example	Meaning
All uppercase plain	SQL> ALTER DATABASE	Indicates command names, SQL reserved words, and keywords.
<i>Italic</i>	Italic is used to indicate a variable: <i>filename</i>	Indicates a value that you must provide. For example, if a command asks you to type <i>filename</i> , you enter the actual name of the file. Italic is also used for emphasis in the text and to indicate the titles of other guides.
square brackets []	x:\[pathname]\oracle\home_name	Encloses optional items. For example, when you create an OFA-compliant Oracle home directory, you can place an optional pathname before the \oracle pathname. Square brackets also indicate a function key, for example [Enter].
C:\>	C:\ORACLE>	Represents the Windows platforms command prompt of the current hard disk drive. Your prompt may differ and may, at times, reflect the subdirectory in which you are working. Referred to as the <i>MS-DOS command prompt</i> in this guide.
Backslash (\) before a directory name	\bin	Indicates that the directory is a subdirectory of the root directory.

Convention	Example	Meaning
<i>oracle_home</i> and <i>oracle_base</i>	Go to the <i>oracle_base\oracle_home\bin</i> directory.	<p>In this Optimal Flexible Architecture (OFA)-compliant release, all subdirectories are no longer under a top level <i>oracle_home</i> directory. There is now a new top-level directory called <i>oracle_base</i> that by default is <i>c:\oracle</i>. The Oracle home directories are located directly under <i>oracle_base</i>.</p> <p>If you install Oracle8i release 8.1.6 on a computer where there is no other Oracle software on the computer, the default settings for the first Oracle home directory is <i>c:\oracle\ora81</i>. If you run Oracle Universal Installer again and install release 8.2.x, the second Oracle home directory is called <i>\ora82</i>.</p> <p>All directory path examples in this guide follow OFA conventions. For more information on OFA, see <i>Oracle8i Administrator's Guide for Windows NT</i>.</p>
<i>HOME_NAME</i>	OracleHOME_NAMETNSListener	Represents the Oracle home name. The home name can be up to sixteen alphanumeric characters. The only special character allowed in the home name is the underscore.
<i>HOMEID</i>	HOME0, HOME1, HOME2	Represents a unique registry subkey for each Oracle home directory in which you install products. A new <i>HOMEID</i> is created and incremented each time you install products to a different Oracle home directory on one machine. Each <i>HOMEID</i> contains its own configuration parameter settings for installed Oracle products.

Convention	Example	Meaning
Symbols	period . comma , hyphen - semicolon ; colon : equal sign = backslash \ single quote ' double quote " parentheses ()	Symbols other than brackets and vertical bars must be entered in commands exactly as shown.

Documentation Library

This guide is part of a larger library of Oracle documentation. The Oracle documentation library consists of two types of documentation:

Documentation Type	Describes...
Operating System-specific	<p>Installation, configuration, and use of Oracle products in a Windows NT or Windows 95/98 environment. Operating system-specific documents are occasionally referred to in the generic documentation set. These documents are easy to identify because they always mention their specific operating system in their title.</p>
Generic	<p>Oracle database, Oracle networking, and Application Programming Interfaces information that is uniform across all operating system platforms. The majority of documents in your documentation set belong to this category. While reading through the generic documentation set, you are occasionally asked to refer to your platform (or operating system) documentation for procedures specific to the Windows NT or Windows 95/98 operating systems.</p> <p>To easily identify where these generic documentation references are described in your operating system documentation, see the index of this guide for the following entry:</p> <p>generic documentation references</p> <p>All generic documentation references described in this guide appear under this index entry.</p>

Related Documents

For more information, see the following manuals.

- *Oracle8i Installation Guide for Windows NT*
- *Oracle8i Release Notes for Windows NT*
- *Oracle8i Administrator's Guide for Windows NT*
- *Oracle Enterprise Manager Administrator's Guide*
- *Oracle Parallel Server Administrator's Guide for Windows NT*
- *Using Microsoft Transaction Server With Oracle8*
- *Net8 Administrator's Guide*
- *Oracle8i Parallel Server Concepts*
- *Getting to Know Oracle8i*
- *Oracle8i Concepts*
- *Oracle8i Reference*
- *Oracle8i Error Messages*

Introducing Oracle COM Automation

This chapter describes the Oracle COM Automation feature Software Development Kit (SDK) and provides an overview of the product. Read this chapter before installing or using Oracle COM Automation feature. Specific topics discussed are:

- [Introducing Oracle COM Automation Feature](#)
- [Benefits of Oracle COM Automation Feature](#)
- [Oracle COM Automation Feature Architecture](#)

Introducing Oracle COM Automation Feature

Component software has been promoted as the next evolution in software development. The growth of object-oriented programming and distributed objects are proof of this industry-wide trend. The scripting of components enables you to reuse code that is pre-built and pre-tested. The reuse of code fulfills one of the key objectives of software development: shorter development cycles and reduced time to market. COM is the ubiquitous technology to promote componentization and reuse of software for Windows-based systems. Oracle COM Automation feature has been created to enable you to use COM-based components to customize and enhance the functionality of the Oracle database on Windows NT.

Updates for Release 8.1.6

Two new demonstration programs have been added:

- [Microsoft PowerPoint Demo](#) - Exchanges data from Oracle to PowerPoint
- [MAPI Demo](#) - Exchanges data from Oracle to Messaging Application Programming Interface (MAPI) compliant applications

See [Chapter 5, "Using Oracle COM Automation Demos"](#) for information on using the demonstration programs.

Additional error messages have been included in this guide. See "[Oracle COM Automation Errors](#)" on page 4-16.

What Is Oracle COM Automation Feature?

Oracle COM Automation feature enables PL/SQL developers to programmatically manipulate COM objects through the OLE Automation interface (IDispatch).

The feature provides a PL/SQL package and exposes a set of application programming interfaces (APIs) to instantiate COM objects, get and set their properties, and invoke their methods. PL/SQL developers can call these APIs from PL/SQL subprograms, stored procedures, stored functions, or triggers to manipulate COM objects. There are no restrictions as to where the COM objects can reside. They can be either local to the database server or be accessed remotely through the Distributed Component Object Model (DCOM).

OLE Automation is the most common and basic mechanism for third-generation language (3GL) and fourth-generation language (4GL) programs to manipulate COM objects. Most COM objects support OLE Automation and the major 4GL programming environments, such as Powersoft PowerBuilder and Microsoft Visual Basic. OLE Automation is the mechanism for scripting COM objects.

Benefits of Oracle COM Automation Feature

Oracle COM Automation feature is a powerful and enabling infrastructure technology for Oracle developers on Windows NT. The feature provides four compelling benefits for developers who are creating and deploying Oracle solutions on Windows NT:

- **Ease of Development**

Oracle COM Automation feature exposes a simple set of APIs to manipulate COM objects. If you are familiar with COM and Microsoft Visual Basic, you will have no problems incorporating these APIs into your PL/SQL subprograms.

- **Reusability**

Oracle COM Automation feature enables you to leverage pre-built COM components that have been developed in-house or by third-party independent software vendors (ISVs). In addition, there are already thousands of pre-existing COM components from which you can choose. The COM component market is expanding rapidly and already offers solutions to many of the most common problems that enterprises need to solve.

- **Flexibility and Extensibility**

You can use Oracle COM Automation feature to customize and enhance the functionality of the Oracle database. Through the use of COM components, the Oracle database can be customized to exchange data between productivity applications, such as Microsoft Word, Microsoft Excel, and Microsoft PowerPoint, generate reports using Crystal Reports, and send and receive e-mail to Mail Application Programming Interface (MAPI) compliant applications. The possibilities for customization and extensibility of the Oracle database are limitless.

- **Enhanced Integration**

Oracle COM Automation feature enables you to deploy Oracle in a Microsoft-centric environment and be assured that Oracle can integrate fully with and capitalize on the services that are exposed by Windows NT, Microsoft BackOffice applications, and Microsoft Office applications.

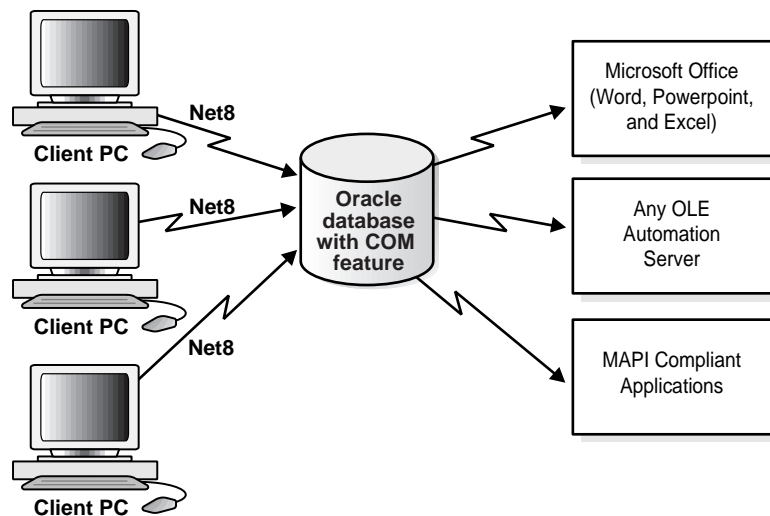
Oracle COM Automation Feature Architecture

Oracle COM Automation feature provides a PL/SQL package of APIs for manipulating COM objects. These APIs are implemented as external procedures in a dynamic linked library (DLL).

Oracle8i supports external procedures which enables developers to call 3GL functions from server-based object type methods and stored procedures. External procedures are invoked exactly like standard PL/SQL stored procedures. However, unlike standard PL/SQL procedures where the body of the procedure is written in PL/SQL and stored in the database, external procedures are C functions that reside within a DLL. You can invoke Oracle COM Automation feature APIs in the same manner as if you are calling a standard PL/SQL stored procedure or function.

Figure 1–1, "Oracle COM Interaction" illustrates the interaction between an Oracle8i database, Oracle COM Automation feature, and external procedures:

Figure 1–1 Oracle COM Interaction



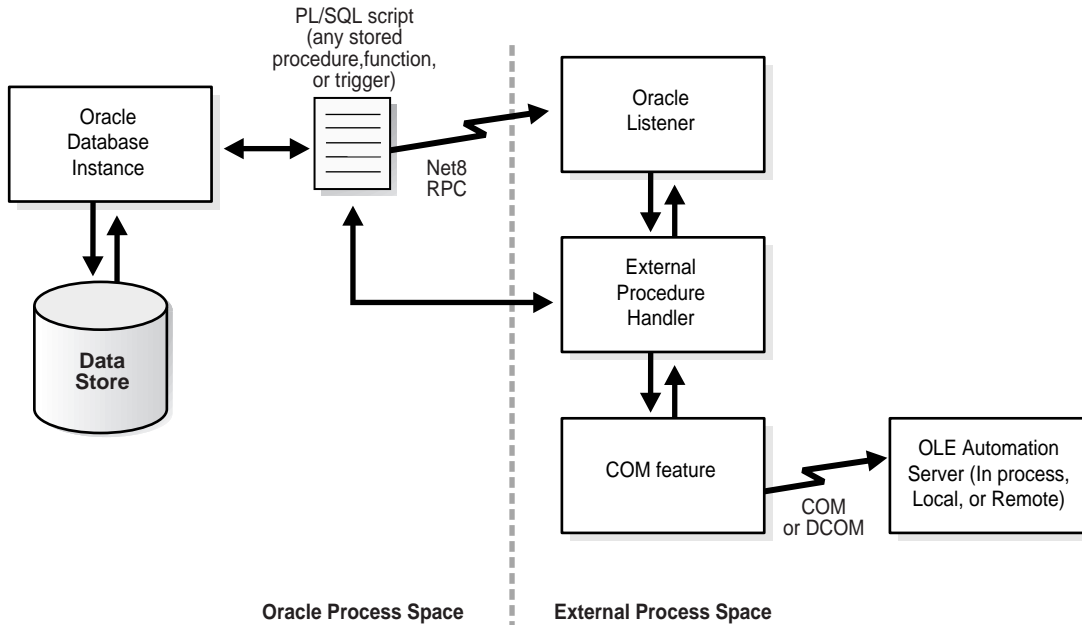
Invoking OLE Automation External Procedure APIs

The Oracle database invokes any of the OLE Automation external procedure APIs as follows:

1. The PL/SQL interpreter looks up the pathname to the Oracle COM Automation feature DLL (`com81.dll`).
2. The PL/SQL interpreter sends a message using Net8 to the listener to start `extproc.exe`, if it has not already been started for the current user session.
3. The PL/SQL interpreter passes the procedure name, parameters, and the pathname of the DLL to `extproc.exe`.
4. `extproc.exe` loads the DLL and executes the external procedure. Each of the OLE Automation external procedure APIs in turn call Win32 APIs that instantiate a COM object, set or get properties of a COM object, or invoke a method of a COM object.
5. `extproc.exe` acts as an intermediary and handles any interaction between Oracle COM Automation feature and Oracle8i database.

Figure 1–2, "Invoking OLE Automation External Procedures" shows an Oracle8i database invoking OLE Automation external procedure APIs.

Figure 1–2 Invoking OLE Automation External Procedures



Architectural Impact on Availability and Performance Issues

The dependence on external procedures by Oracle COM Automation feature has implications for the availability of the Oracle8i database and the performance of Oracle COM Automation feature.

Oracle Database Availability

You do not jeopardize the availability of the Oracle database by using Oracle COM Automation feature and custom or third-party COM objects in a production environment. Oracle COM Automation feature operates outside of the Oracle kernel's address space. This safeguards the Oracle kernel from COM objects that crash unexpectedly.

Oracle COM Automation Feature Performance

Currently, there are two architectural models in which an external procedure can be invoked. The model used directly affects the performance of Oracle COM Automation feature. In this release, you can choose Architectural Model 1 or Architectural Model 2.

- Architectural Model 1

A separate process, EXTPROC, is responsible for loading external DLLs and handling the interaction between the Oracle kernel and Oracle COM Automation feature. A copy of EXTPROC is spawned by the Oracle kernel when a user makes the first OLE Automation external procedure API call. EXTPROC remains in memory until the user's session terminates. Therefore, there will be a separate EXTPROC process for each user session that makes an external procedure call.

The first external procedure call incurs substantial overhead because Oracle needs to spawn a new EXTPROC process. Process creation on Windows NT is much slower than thread creation. However, subsequent external procedure calls in the same session will execute rapidly. If you use this model, you should consolidate as many external procedure calls as possible in one session to achieve optimum performance.

- Architectural Model 2

EXTPROC operates as an NT process that is capable of spawning multiple threads. When multiple sessions are connected to an Oracle database, only one EXTPROC process is spawned by Oracle instead of spawning separate EXTPROC processes to handle each database session. This one EXTPROC process in turn spawns one thread per database session.

Installing Oracle COM Automation

This chapter provides an overview of the Oracle COM Automation installation. The following topics are included:

- [Oracle COM Automation Feature Components](#)
- [System Requirements](#)

See Also: For installation instructions, see the *Oracle8i Installation Guide for Windows NT*.

Oracle COM Automation Feature Components

The Oracle COM Automation package contains the features and demos that illustrate how to use this product to solve real-world problems. It includes the following components:

- Oracle COM Automation feature (`com81.dll`)
- PL/SQL installation and definition script (`comwrap.sql`).
- Oracle COM Automation demonstration programs

All the components, except `com81.dll`, are located in the `oracle_base\oracle_home\com` directory

System Requirements

Oracle COM Automation feature requires an Oracle8i database and Windows NT 4.0. Before you install Oracle COM Automation feature, you must have a functioning Oracle8i database on the computer.

To use the Oracle COM Automation feature demonstrations discussed in [Chapter 5, "Using Oracle COM Automation Demos"](#), you need to first install the application that is used in the demonstration program. The application must be installed before you run any of demonstration scripts.

Post-Installation Configuration Tasks

This chapter describes post-installation configuration tasks for Oracle COM Automation feature. Specific topics discussed are:

- [Optimal Flexible Architecture for Oracle8i](#)
- [Configuring Oracle COM Automation Feature](#)
- [Configuring the Listener](#)
- [Configuring DCOM](#)

Optimal Flexible Architecture for Oracle8i

When you install Oracle8i Enterprise Edition utilizing the Optimal Flexible Architecture (OFA), all subdirectories are under a top-level directory called *oracle_base* that is of the form *x:\oracle* where *X* is any hard drive. If you install an OFA-compliant database using Oracle Universal Installer defaults, *oracle_base* is *c:\oracle*. *\oracle_home* directories are located under *oracle_base*. *\oradata* and *\admin* directories that contains the database files and database administration files are also located under *oracle_base*.

The following configuration instructions reference directory trees that assume that you have installed an OFA-compliant Oracle database.

Configuring Oracle COM Automation Feature

To configure Oracle COM Automation feature:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com
```

where *oracle_base\oracle_home* represents your drive letter and the Oracle home directory where Oracle COM Automation feature was installed.

2. Start Server Manager:

```
C:\> svrmgr1
```

3. Connect to the Oracle database instance as SYSTEM.

```
SVRMGR> CONNECT SYSTEM/PASSWORD[@DB_ALIAS]
```

4. Grant the CREATE LIBRARY privilege to the database users that will use Oracle COM Automation feature. For example:

```
SVRMGR> GRANT CREATE LIBRARY TO SCOTT;
```

5. Connect to the user that will use Oracle COM Automation and run the *comwrap.sql* script at the Server Manager prompt:

```
SVRMGR> CONNECT SCOTT/TIGER  
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\COMWRAP.SQL
```

You will receive several “ORA-04043: object XXXX does not exist” messages when you run this script for the first time. These messages are normal.

Configuring the Listener

Oracle COM Automation feature relies on external procedure callouts and you must configure the listener and Net8 remote procedure call (RPC) mechanism for the feature to work.

The following examples demonstrate how to configure the `listener.ora` and `tnsnames.ora` files to use inter-process communication (IPC) to invoke external stored procedures. This is the default configuration for both files for this release.

See Also: For additional information on how to configure the `listener.ora` and `tnsnames.ora` files for external procedures, see the *Net8 Administrator's Guide*.

listener.ora Configuration File

```

LISTENER =
(ADDRESS_LIST =
  (ADDRESS=
    (PROTOCOL= IPC)
    (KEY= EXTPROC0)
  )
)
STARTUP_WAIT_TIME_LISTENER = 0
CONNECT_TIMEOUT_LISTENER = 10
TRACE_LEVEL_LISTENER = OFF
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = ORCL)
    )
    (SID_DESC =
      (SID_NAME = extproc)
      (PROGRAM=extproc)
    )
  )
)
PASSWORDS_LISTENER = (oracle)

```

tnsnames.ora Configuration File

```

extproc_connection_data.world =
  (DESCRIPTION =
    (ADDRESS =
      (PROTOCOL=IPC)
    )
  )
)

```

```
)  
(CONNECT_DATA = (SID=extproc)  
)  
)
```

ORA-28575: Error Message

An “ORA-28575: unable to open RPC connection to external procedure agent” error message indicates one of two possible problems:

Problem 1

Problem: The listener is not started.

Action: You must start the Oracle`home_name`TNSListener service from the Control Panel or the MS-DOS command prompt.

To start Oracle services from the Control Panel:

1. Choose Start > Settings > Control Panel.

The *Control Panel* window appears.

2. Double-click Services.

The *Services* dialog box appears.

3. Find Oracle`HOME_NAME`TNSListener in the list and verify that it has a status of *Started*. If it does not, select it and click Start.

To start Oracle services from the MS-DOS command prompt:

Enter the following command to start an Oracle service at the MS-DOS command prompt:

```
C:\> net start service
```

where *service* is a specific service name, such as Oracle`HOME_NAME`TNSListener.

Problem 2

Problem: The listener is not configured correctly.

Action: You must modify the `listener.ora` and `tnsnames.ora` files. See ["Configuring the Listener"](#) on page 3-3 for information on how to configure these files.

Configuring DCOM

Oracle COM Automation feature supports the use of DCOM to access remote COM objects over a network. However, in order to use DCOM, you must configure both the computer that is running the Oracle database instance and the computer that contains the remote COM object.

For security purposes, the Oracle listener must run with the same security privileges as a domain user that has access to the remote computer. In order to authenticate that the client has access to the remote computer, DCOM passes the security credentials of the Oracle listener to the remote computer. The remote computer validates the security credentials and allows DCOM to proceed. Normally, the Oracle listener runs as the System account rather than a specific user account. Because the System account has no remote privileges, the system administrator must perform the following steps.

To change the Oracle listener to run as a domain user:

1. Choose Start > Settings > Control Panel. The *Control Panel* window appears.
2. Double-click Services. The *Services* dialog box appears.
3. Select the OracleHOME_NAMETNSListener service and click Startup.
4. Click the This Account radio button. The *Service* dialog box appears.
5. Enter the name or browse for a domain user. The Oracle listener runs with the security privileges of this user.
6. Enter and confirm the password of the domain user that was selected.
7. Click OK to save the changes.

Note: After you change the Oracle listener's security privilege to a domain user, you must also change the security privilege of the Oracle database to the same domain user or a domain user that has the compatible security privilege; otherwise, no one will be able to connect to Oracle.

The next step is to configure the DCOM security settings of the remote computer. The system administrator must set the DCOM security privileges such that the Oracle listener, operating as a domain user, has sufficient privileges to instantiate and manipulate the remote COM object.

The remote COM object executes with the same privileges as the Oracle listener. If the COM object attempts to perform an action for which it does not have permission, DCOM denies the operation and returns a security violation back to Oracle COM Automation feature. It is essential that the system administrator configure the DCOM security properly and provide Oracle with the necessary permissions.

DCOM enables the administrator to configure the default security for the entire computer or define specific security permissions for a specific COM object. Microsoft provides the `dcomcnfg.exe` tool to configure DCOM security. This tool enables the system administrator to set the access permissions, launch permissions, and configuration permissions for a specific COM object or all COM objects on a computer. For more information on how to use this tool and the implications of each of these permissions, see the Microsoft operating system specific documentation for the computer.

Oracle COM Automation Core Functionality

This chapter describes the core functionality of Oracle COM Automation feature. Specific topics discussed are:

- [Oracle COM Automation Feature Core Functionality](#)
- [Developing Solutions Using Oracle COM Automation](#)
- [PL/SQL Application Programming Interfaces](#)
- [Oracle COM Automation Errors](#)
- [OLE Automation Errors](#)

Oracle COM Automation Feature Core Functionality

Oracle COM Automation feature provides a mechanism to manipulate COM objects. It acts as a generic wrapper interface of the IDispatch interface (OLE Automation).

- The feature externalizes all the methods supported by the IDispatch interface.
- COM objects expose properties, data attributes, and methods, functions that perform an action, to the developer.
- The IDispatch interface supports three basic operations for any COM object:
 - Get the value of an exposed property.
 - Set the value of an exposed property.
 - Invoke a method on an object.

When an Oracle COM Automation feature API is invoked from PL/SQL, the feature converts the parameters to the appropriate OLE Automation data types and then invokes the corresponding IDispatch API with the converted parameters. The feature externalizes the following APIs:

- [CreateObject](#)
- [DestroyObject](#)
- [GetLastError](#)
- [GetProperty](#)
- [SetProperty](#)
- [InitArg](#)
- [SetArg](#)
- [SetPtrArg](#)
- [Invoke](#)

This section describes each of the APIs and how to use them in PL/SQL blocks. A typical PL/SQL block performs the following steps to create and manipulate a COM object using Oracle COM Automation feature:

1. Call [CreateObject](#) to create the COM object.

2. Manipulate the COM object using the following APIs:
 - Call `GetProperty` to get a property value.
 - Call `SetProperty` to set a property value to a new value.
3. Call `Invoke` to call a method. As part of preparation for the `Invoke` API call, you use `InitArg`, `SetArg`, and `SetPtrArg` to package the argument to be sent to the OLE Automation method.
4. Call `GetLastError` to get the most recent error information.
5. Call `DestroyObject` to destroy the object.

Developing Solutions Using Oracle COM Automation

Oracle COM Automation feature enables you to use components that provide additional functionality that PL/SQL does not support. You can choose to build your own custom components or use the thousands of pre-built components that are available from third-party ISVs.

Information Required for COM Objects

Before you can begin building a solution using Oracle COM Automation feature, you must know two things about the COM objects that you intend to use.

1. You must determine the Program ID of the COM object. The Program ID, or `progID`, is a descriptive string that maps to the Globally Unique Identifier (GUID), which is a hexadecimal number that uniquely identifies a COM object. An example of a `progID` is the following string:

```
Excel.Worksheet.1
```

Use the `progID` to tell the `CreateObject` API which COM object to instantiate.

2. You must be aware of the types of properties and methods that are exposed through the COM object's `IDispatch` interface. Usually, the ISV provides documentation describing the names and datatype of the object's properties and the prototypes of the object's methods. Properties are referred to by a descriptive string, such as *xpos* or *ypos*. A property can be any standard OLE Automation datatype, such as integer or string. The `GetProperty` and `SetProperty` APIs take the property name and a variable of the appropriate datatype. Methods are referred to by a descriptive string, such as *InsertChart*. A method takes a set of parameters that are of different OLE Automation datatypes and returns an OLE Automation datatype.

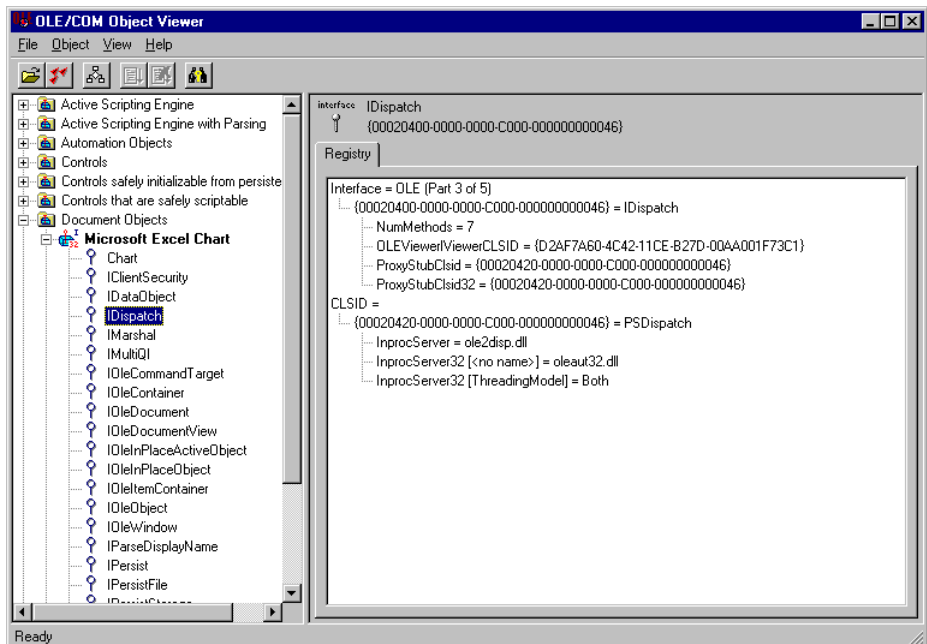
The following is an example of an OLE Automation method prototype in Interface Definition Language (IDL) grammar:

```
[id(0x6003000)]
long Post([in, out] long* lngAccountNo,
          [in, out] long* lngAmount,
          [in, out] BSTR* strResult);
```

OLE/COM Object Viewer

Microsoft provides a tool called the OLE/COM Object Viewer with Microsoft Visual C++ for browsing the properties and methods of COM objects on a local system. This tool enables you to quickly and easily determine the properties and methods that each COM object exposes.

Figure 4–1 OLE/COM Object Viewer



Datatype Conversion

Because Oracle uses PL/SQL datatypes and OLE Automation uses Microsoft Visual Basic datatypes, Oracle COM Automation feature must convert the data that it receives from PL/SQL and pass it to the OLE Automation object, and vice versa.

[Table 4–1, "PL/SQL to Visual Basic Datatypes"](#) shows the conversion from PL/SQL datatypes to Microsoft Visual Basic datatypes:

Table 4–1 PL/SQL to Visual Basic Datatypes

PL/SQL Datatype	Microsoft Visual Basic Datatype
Varchar2	String
Boolean	Boolean
Binary_Integer	Byte, Integer, or Long
Double Precision	Double, Single, or Currency
Date	Date

[Table 4–2, "Visual Basic to PL/SQL Datatypes"](#) shows the conversion from Microsoft Visual Basic datatypes to PL/SQL datatypes:

Table 4–2 Visual Basic to PL/SQL Datatypes

Microsoft Visual Basic Datatype	PL/SQL Datatype
Boolean	Boolean
Long, Integer, Byte, Object	Binary_Integer
String	Varchar2
Double, Single, or Currency	Double Precision
Date	Date

HRESULT Return Codes

These APIs return an integer return code. The return code is 0 when successful or a non-zero HRESULT when an error occurs. An HRESULT is an OLE error code of the hexadecimal form `0x800nnnnn`, but when it is returned as a `binary_integer` value, it has the form `-214nnnnnn`. For example, passing an invalid object name to `CreateObject` causes it to return a `binary_integer` HRESULT of `-2147221005`, which is `0x800401f3` in hexadecimal.

See "[GetLastError](#)" on page 4-8 and "[OLE Automation Errors](#)" on page 4-18 for additional information on how to interpret the return codes from Oracle COM Automation feature. For complete information on HRESULTs, refer to the Microsoft documentation on HRESULTs.

PL/SQL Application Programming Interfaces

The following section describes the PL/SQL APIs for manipulating COM objects using the OLE Automation interface. Each of the following PL/SQL stored procedures reside in the package `ORDCOM`.

CreateObject

Instantiates a COM object in an OLE Automation server.

Syntax

```
FUNCTION CreateObject(progid VARCHAR2, reserved BINARY_INTEGER, servername
VARCHAR2, objecttoken OUT BINARY_INTEGER) RETURN BINARY_INTEGER;
```

where:

progid is the programmatic identifier (progID) of the OLE Automation object to create. This character string describes the class of the OLE Automation object and has the following form:

OLEComponent.Object

OLEComponent is the component name of the OLE Automation server, and *Object* is the name of the OLE Automation object. The specified OLE Automation object must be creatable and must support the IDispatch interface.

reserved	Currently, this parameter is reserved for future use. Pass a value of 0. Future versions of Oracle COM Automation feature may use this parameter.
servername	is the name of the remote DCOM server on which to instantiate the COM object. Passing a specified name forces Oracle COM Automation feature to attempt to instantiate the COM object on a remote computer. Passing an empty string, for example, "", forces Oracle COM Automation feature to check the registry for the location of the COM object. The registry contains information as to whether the COM object is local or remote. Therefore, to create a local COM object, always pass an empty string and ensure the registry indicates that the COM object exists locally. The registry information for COM objects can be configured with the tool DCOMCNFG.EXE.
objecttoken	is the returned object token. It must be a local variable of datatype binary_integer. This object token identifies the created OLE Automation object and is used in calls to the other Oracle COM Automation feature APIs.

Remarks

The created OLE Automation object is freed with a corresponding call to **DestroyObject**. This nullifies the internal representation of the object in the Oracle COM Automation Feature and releases all the interfaces associated with the object.

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Code Sample

```
hresult binary_integer;
applicationToken binary_integer:=-1;

hresult :=ORDCOM.CreateObject('Excel.Application', 0, '', applicationToken);
IF hresult = -1 THEN
    dbms_output.put_line(hresult);
END IF;
```

DestroyObject

Destroys a created OLE Automation object.

Syntax

```
FUNCTION DestroyObject(objecttoken BINARY_INTEGER) RETURN BINARY_INTEGER;
```

where:

objecttoken is the object token of an OLE Automation object previously created by **CreateObject**.

Remarks

Calling **DestroyObject** nullifies the internal representation of the object in the Oracle COM Automation Feature and releases all the interfaces associated with the object.

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Code Sample

```
hresult binary_integer;
applicationToken binary_integer:=-1;

/*
  At some point before this, we called CreateObject and
  got a valid applicationToken.
*/
hresult:=ORDCOM.DestroyObject(applicationToken);
```

GetLastError

Obtains the OLE Automation error information about the last error that occurred.

Syntax

```
FUNCTION GetLastError(source OUT VARCHAR2, description OUT VARCHAR2, helpfile
OUT VARCHAR2, helpid OUT BINARY_INTEGER) RETURN BINARY_INTEGER;
```

where:

source is the source of the error information. If specified, it must be a local *char* or *varchar* variable. The return value is truncated to fit the local variable if necessary.

<i>description</i>	is the description of the error. If specified, it must be a local <i>char</i> or <i>varchar</i> variable. The return value is truncated to fit the local variable if necessary.
<i>helpfile</i>	is the Help file for the OLE Automation object. If specified, it must be a local <i>char</i> or <i>varchar</i> variable. The return value is truncated to fit the local variable if necessary.
<i>helpid</i>	is the Help file context ID. If specified, it must be a local <i>int</i> variable.

Remarks

Each call to an Oracle COM Automation feature API (except **GetLastError**) resets the error information, so that **GetLastError** obtains error information only for the most recent Oracle COM Automation feature API call. Because **GetLastError** does not reset the last error information, it can be called multiple times to get the same error information.

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

See "[OLE Automation Errors](#)" on page 4-18 for a description of the types of errors that can be returned by this function.

Code Sample

```

applicationToken binary_integer:= -1;
hresult binary_integer;
error_src varchar2(255);
error_description varchar2(255);
error_helpfile varchar2(255);
error_helpID binary_integer;

hresult:=ORDCOM.CreateObject('Excel.Application', 0, '', applicationToken);
IF hresult=-1 THEN
    ORDCOM.GetLastError(error_src, error_description, error_helpfile, error_
helpID);
    dbms_output.put_line(error_src);
    dbms_output.put_line(error_description);
    dbms_output.put_line(error_helpfile);
    return hresult;
END IF;

```

GetProperty

Gets a property value of an OLE Automation object.

Syntax

```
FUNCTION GetProperty(objecttoken BINARY_INTEGER, propertyname VARCHAR2, argcount  
BINARY_INTEGER, propertyvalue OUT any PL/SQL datatype) RETURN BINARY_INTEGER;
```

where:

<i>objecttoken</i>	is the object token of an OLE object previously created by CreateObject .
<i>propertyname</i>	is the property name of the OLE object to return.
<i>argcount</i>	is the index of the property array. If the property is not an array, then the developer should specify 0.
<i>propertyvalue</i>	is the returned property value. The returned property type depends on the OLE Automation type that is returned. You must pass the PL/SQL datatype that corresponds to the Microsoft Visual Basic datatype of the OLE Automation property. Otherwise, the OLE Automation feature will not properly convert the Microsoft Visual Basic datatype.
any PL/SQL datatype	supported by COM Automation Feature.

Remarks

If the property returns an OLE object, you must specify a local variable of datatype `binary_integer` or the *propertyvalue* parameter. An object token is stored in the local variable, and this object token can be used with other OLE Automation stored procedures.

When the property returns an array, if *propertyvalue* is specified, it is set to NULL.

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Code Sample

```
ChartObject binary_integer := -1;  
ChartToken binary_integer := -1  
hresult binary_integer;  
/* Previously, ChartObject, ChartToken were initialized calling CreateObject */  
hresult := ORDCOM.GetProperty(ChartObject, 'Chart', 0, ChartToken);
```

```

IF hresult=-1 THEN
    /* Do error checking here */
    return hresult;
END IF;

```

SetProperty

Sets a property of an OLE Automation object to a new value.

Syntax

```

FUNCTION SetProperty(objecttoken BINARY_INTEGER, propertyname VARCHAR2, newvalue
ANY PL/SQL DATATYPE, datatype VARCHAR2) RETURN BINARY_INTEGER;

```

where:

<i>objecttoken</i>	is the object token of an OLE Automation object previously created by CreateObject .
<i>propertyname</i>	is the property name of the OLE object to set to a new value.
<i>newvalue</i>	is the new value of the property. It must be a value of the appropriate datatype.
<i>datatype</i>	explicitly specifies the datatype of the value passed in. The list of available datatypes are: <ul style="list-style-type: none"> ▪ I2 - 2 byte integer ▪ I4 - 4 byte integer ▪ R4 - IEEE 4 byte real ▪ R8 - IEEE 8 byte real ▪ SCODE - error code ▪ CY - currency ▪ DISPATCH - dispatch pointer ▪ BSTR - String ▪ BOOL - boolean ▪ DATE - date

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Code Sample

```
RangeToken  binary_integer:=-1;
hresult     binary_integer;

/*
   Previously, RangeToken has been initialized to a valid object token with a
   property by the name of value.
*/
hresult:=ORDCOM.SetProperty(RangeToken, 'Value', 'EmpNo', 'BSTR');
IF hresult=-1 THEN
   /* Do error checking here */
   return hresult;
END IF;
```

InitArg

Initializes the parameter set to pass to an Invoke call.

Syntax

```
PROCEDURE InitArg();
```

Remarks

Each *SetArg* or *SetPtrArg* procedure sets the *n*th parameter value. The *InitArg* call initializes the parameter set. After *InitArg* has been called, a *SetArg* or *SetPtrArg* call sets the first parameter to the specified value. A second *SetArg* or *SetPtrArg* call sets the second parameter in the parameter list. Subsequent calls set the *n*th parameters in the parameter list, where *n* is the number of times *SetArg* or *SetPtrArg* has been called after an *InitArg* call. Another call to *InitArg* resets the argument list and a call to *SetArg* or *SetPtrArg* sets the first parameter again.

Code Sample

See "[Invoke](#)" on page 4-15 for sample code.

SetArg

Used to construct the parameter list for the next *Invoke* call. *SetArg* sets a parameter's value to be passed by value.

Syntax

```
PROCEDURE SetArg(paramvalue ANY PL/SQL DATATYPE, datatype VARCHAR2);
```

where:

paramvalue is the value of the parameter to be passed to an *Invoke* call. The parameter set is the *n*th parameter in the parameter list, where *n* is the numbers of times *SetArg* or *SetPtrArg* has been called after an *InitArg* call.

datatype explicitly specifies the datatype for the value passed in. The list of available datatypes are:

- I2 - 2 byte integer
- I4 - 4 byte integer
- R4 - IEEE 4 byte real
- R8 - IEEE 8 byte real
- SCODE - error code
- CY - currency
- DISPATCH - dispatch pointer
- BSTR - String
- BOOL - boolean
- DATE - date

Remarks

Each *SetArg* or *SetPtrArg* procedure sets the *n*th parameter value. The *InitArg* call initializes the parameter set. After *InitArg* has been called, a *SetArg* or *SetPtrArg* call sets the first parameter to the specified value. A second *SetArg* or *SetPtrArg* call sets the second parameter in the parameter list. Subsequent calls set the *n*th parameters in the parameter list, where *n* is the number of times *SetArg* or *SetPtrArg* has been called after an *InitArg* call. Another call to *InitArg* resets the argument list and a call to *SetArg* or *SetPtrArg* sets the first parameter again.

Code Sample

See "[Invoke](#)" on page 4-15 for sample code.

SetPtrArg

Constructs the parameter list for the next *Invoke* call. *SetPtrArg* sets a parameter's value to be passed by reference.

Syntax

```
PROCEDURE SetPtrArg(paramvalue ANY PL/SQL DATATYPE, datatype VARCHAR2);
```

where:

paramvalue is the value of the parameter to be passed to an *Invoke* call. The parameter set is the *n*th parameter in the parameter list, where *n* is the numbers of times *SetArg* or *SetPtrArg* has been called after an *InitArg* call.

datatype explicitly specifies the datatype for the value passed in. The list of available datatypes are:

- I2 - 2 byte integer
- I4 - 4 byte integer
- R4 - IEEE 4 byte real
- R8 - IEEE 8 byte real
- SCODE - error code
- CY - currency
- DISPATCH - dispatch pointer
- BSTR - String
- BOOL - boolean
- DATE - date

Remarks

Each *SetArg* or *SetPtrArg* procedure sets the *n*th parameter value. The *InitArg* call initializes the parameter set. After *InitArg* has been called, a *SetArg* or *SetPtrArg* call sets the first parameter to the specified value. A second *SetArg* or *SetPtrArg* call sets the second parameter in the parameter list. Subsequent calls set the *n*th parameters in the parameter list, where *n* is the number of times *SetArg* or *SetPtrArg* has been called after an *InitArg* call. Another call to *InitArg* resets the argument list and a call to *SetArg* or *SetPtrArg* sets the first parameter again.

Code Sample

See ["Invoke"](#) on page 4-15 for sample code.

Invoke

Calls a method of an OLE Automation object. This function uses the parameter list, previously created by the calls to `InitArg`, `SetArg`, and `SetPtrArg`, as input for the OLE Automation method.

Syntax

```
FUNCTION Invoke(objecttoken BINARY_INTEGER, methodname VARCHAR2, argcount
BINARY_INTEGER, returnvalue OUT ANY PL/SQL DATATYPE) RETURN BINARY_INTEGER;
where:
```

<i>objecttoken</i>	is the object token of an OLE Automation object previously created by CreateObject .
<i>methodname</i>	is the method name of the OLE Automation object to call.
<i>argcount</i>	is the number of arguments passed to the OLE Automation object method.
<i>returnvalue</i>	is the return value of the method of the OLE Automation object. If specified, it must be a local variable of the appropriate datatype.

Remarks

If the method's return value is an OLE object, then the developer must specify a local variable of datatype `binary_integer` for the *returnvalue* parameter. An object token is stored in the local variable, and this object token can be used with other Oracle COM Automation feature APIs.

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Code Sample

```
xpos binary_integer;
ypos binary_integer;
width binary_integer;
height binary_integer;
WorkSheetToken binary_integer:=-1;
ChartObjectToken binary_integer:=-1;
ChartObject binary_integer:=-1;
hresult binary_integer;

/*WorkSheetToken has been initialized with a valid OLE Automation object */
```

```
/* Executes a method that takes 0 arguments */
ORDCOM.InitArg();
i:=ORDCOM.Invoke(WorkSheetToken, 'ChartObjects', 0, ChartObjectToken);

/* Executes a method that takes 4 arguments */
ORDCOM.InitArg();
ORDCOM.SetArg(xpos, 'I2');
ORDCOM.SetArg(ypos, 'I2');
ORDCOM.SetArg(width, 'I2');
ORDCOM.SetArg(height, 'I2');
i:=ORDCOM.Invoke(ChartObjectToken, 'Add', 4, ChartObject);
```

Oracle COM Automation Errors

The following is a list of Oracle COM Automation errors and their common causes.

COM-0001: Not a boolean type

Action: Ensure that the variable is of the appropriate datatype.

COM-0002: Invalid Token or no interface for token

Action: Ensure that the interface exists.

COM-0003: Maximum Objects reached

Action: Make sure that objects are destroyed after they are used by calling [DestroyObject](#).

COM-0004: The registered CLSID for the ProgID is invalid.

Action: Check that the COM component of the specified ProgID is registered.

COM-0005: An error occurred writing the CLSID to the registry.

Action: Make sure your registry can be written to and is not corrupted.

COM-0006: A specified class is not registered in the registration database

Action: Make sure the class is registered.

COM-0007: Failed to initialize OLE Automation object

Action: Make sure the object is registered as a COM Automation object.

COM-0008: No interface is supported

Action: Check that the interface specified is valid.

COM-0009: Failed to get type info count

Action: Check that the object is properly registered.

COM-0010: Does not support type info implementation

Action: Check that the object is properly registered.

COM-0011: Failed to get type information

Action: Check that the object is properly registered.

COM-0012: Failed to get type attributes.

Action: Check that the object is properly registered.

COM-0013: Failed to get function description at index.

Action: Check that the object is properly registered.

COM-0014: Failure to invoke

Action: Check that the method name is valid for the object.

COM-0015: Bad parameter count

Action: Make sure the number of parameters for a method is equal to the count.

COM-0016: One of the arguments in rgvarg is not a valid variant type.

Action: Check that the object is properly registered.

COM-0017: The application needs to raise an exception. The structure passed in pexcepinfo should be filled in.

Action: Make sure structure in pexcepinfo is initialized.

COM-0018: The requested member does not exist, or the call to Invoke tried to set the value of a read-only property.

Action: Make sure the property value can be written to or the member exists.

COM-0019: This implementation of IDispatch does not support named arguments.

Action: Do not use named arguments. Use standard parameter passing.

COM-0020: One of the arguments in rgvarg could not be coerced to the specified type.

Action: Make sure that the coerced arguments are of compatible data types.

COM-0021: One of the parameter dispatch IDs does not correspond to a parameter on the method.

Action: Make sure the arguments are passed in correctly.

COM-0022: One or more of the arguments could not be coerced.

Action: Make sure your arguments are compatible.

COM-0023: The interface ID passed in riid is not IID_NULL.

Action: Make sure the interface ID passed is IID_NULL.

COM-0024: The member being invoked interprets string arguments according to an unrecognized locale ID (LCID).

Action: Make sure your localeID is valid.

COM-0025: Not an optional parameter

Action: Make sure your argument count is correct for the number of parameters passed in.

COM-0026: Name exceeded the maximum character allowed

Action: Enter 1024 characters or less for the name.

COM-0027: This class cannot be created as part of an aggregate.

Action: Do not create this class as part of an aggregate.

OLE Automation Errors

The following is a list of OLE Automation errors and their common causes. Both the hexadecimal and binary error codes are listed.

(0x800401f3) (-2147221005) Invalid class string

Cause: The specified ProgID or CLSID is not registered as an OLE object in the registry of the local computer.

(0x8007007e) (-2147024770) The specified module could not be found

Cause: The specified OLE object is registered as an in-process OLE server (.DLL file), but the .DLL file could not be found or loaded.

(0x80020004) (-2147352572) Parameter not found

Cause: A named parameter was specified before a positional parameter.

Action: Ensure that all named parameters are specified after all positional parameters.

(0x80020005) (-2147352571) Type mismatch

Cause: The datatype of a PL/SQL local variable used to store a returned property value or a method return value did not match the Visual Basic data type of the property or method return value.

Action: Ensure that the local variable is of the appropriate datatype. Or, the return value of a property or a method was requested, but it does not return a value.

(0x80020006) (-2147352570) Unknown name

Cause: The specified property or method name was not found for the specified object.

(0x80020008) (-2147352568) Bad variable type

Cause: The datatype of a PL/SQL value passed as a method parameter did not match the Microsoft Visual Basic data type of the method parameter, or a NULL value was passed as a method parameter.

Action: Ensure that any local variables used as method parameters are of the appropriate datatype and are set to a value other than NULL.

(0x80080005) (-2146959355) Server execution failed

Cause: The specified OLE object is registered as a local OLE server (.EXE file), but the .EXE file could not be found or started.

Using Oracle COM Automation Demos

This chapter describes how to use Oracle COM Automation feature demonstration programs. Specific topics discussed are:

- [Oracle COM Automation Feature Demos Overview](#)
- [Microsoft Word Demo](#)
- [Microsoft Excel Demo](#)
- [Microsoft PowerPoint Demo](#)
- [MAPI Demo](#)

Oracle COM Automation Feature Demos Overview

Oracle COM Automation feature includes demos to give you an idea of how to use Oracle COM Automation feature to build solutions. These demos provide base functionality. They are provided as examples of how to use Oracle COM Automation feature and as a foundation upon which to build more customized, complex applications that use OLE Automation.

Each demo exposes a core set of APIs that enables you to do simple operations using OLE Automation. Each OLE Automation server, such as Word and Excel, provides more advanced capabilities than what is offered through the demo APIs. To take advantage of these advanced features, you must design and code your own PL/SQL procedures.

In this release, Oracle Corporation has provided the following demos:

- [Microsoft Word Demo](#) - Exchanges data from Oracle to Word
- [Microsoft Excel Demo](#) - Exchanges data from Oracle to Excel
- [Microsoft PowerPoint Demo](#) - Exchanges data from Oracle to PowerPoint
- [MAPI Demo](#) - Exchanges data from Oracle to Messaging Application Programming Interface (MAPI) compliant applications

Microsoft Word Demo

The following sections describe how to install the Microsoft Word demo and the APIs that it exposes. This demo is provided as an example of the types of solutions that can be built with Oracle and Microsoft Word.

The Microsoft Word demo provides a PL/SQL package (ORDWord) that exposes several APIs for manipulating Microsoft Word. Also, the Microsoft Word demo includes a script to demonstrate the capabilities of exchanging data between Oracle and Microsoft Word. The `worddem.sql` script exchanges data from the EMP table in Oracle to a Microsoft Word document.

Installing the Microsoft Word Demo

Microsoft Word must be installed on the local computer before installing this demo.

To install the Microsoft Word demo, perform the following steps:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com\demos
```

2. Start Server Manager:

```
C:\> svrmgr1
```

3. Connect to the Oracle database instance as the user that will use the Microsoft Word demo. For example, SCOTT/TIGER.

```
SVRMGR> CONNECT SCOTT/TIGER
```

4. Run the `wordsol.sql` script at the Server Manager prompt:

```
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\DEMOS\WORDSOL.SQL
```

This script creates the `ORDWord` package in the current user's schema. You will receive several `ORA-04043: object XXXX does not exist` when you execute this script for the first time. These errors are normal.

Using the Microsoft Word Demo

To run the Microsoft Word demo, perform the following steps:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com\demos
```

2. Start Server Manager:

```
C:\> svrmgr1
```

3. Connect to the Oracle database instance as the user that will use the Microsoft Word demo. For example, SCOTT/TIGER.

```
SVRMGR> CONNECT SCOTT/TIGER
```

4. Run the `worddem.sql` script at the Server Manager prompt:

```
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\DEMOS\WORDDEM.SQL
```

This script creates a Microsoft Word document (`worddemo.doc`) in the `C:\` directory. The document contains data from the `EMP` table.

5. Open `worddemo.doc` to see its contents.

Core Functionality

The following subsections describe the APIs that the Microsoft Word demo exposes. These APIs are primitive. Be aware that much of the functionality that Microsoft Word exposes through OLE Automation is not exposed through these APIs. These APIs and PL/SQL code are provided as a *proof of concept* that Oracle COM Automation feature is a viable development feature.

CreateWordObject

Instantiates a 'Word.Basic' object in the Microsoft Word Automation server.

Syntax

```
FUNCTION CreateWordObject() RETURN BINARY_INTEGER;
```

Remarks

This function must be called before any other operation can be performed. This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

FileNew

Creates a new Microsoft Word document.

Syntax

```
FUNCTION FileNew() RETURN BINARY_INTEGER;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

FileLoad

Loads a document into Microsoft Word.

Syntax

```
FUNCTION FileLoad(filename VARCHAR2) RETURN BINARY_INTEGER;
```

where:

filename is the fully qualified file name of the document.

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

FileSave

Saves the current active Microsoft Word document to disk.

Syntax

```
FUNCTION FileSave() RETURN BINARY_INTEGER;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

FileSaveAs

Saves the current active Microsoft Word document as a specific file.

Syntax

```
FUNCTION FileSaveAs(filename VARCHAR2) RETURN BINARY_INTEGER;
```

where:

filename is the fully qualified file name of the document.

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

FileClose

Closes the current active Microsoft Word document.

Syntax

```
FUNCTION FileClose() RETURN BINARY_INTEGER;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

InsertText

Inserts a text string into the current active Microsoft Word document.

Syntax

FUNCTION **InsertText**(*textstr* VARCHAR2) RETURN BINARY_INTEGER;

where:

textstr is the text that will be inserted into the document.

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

InsertNewLine

Inserts a carriage return into the current active Microsoft Word document.

Syntax

FUNCTION **InsertNewLine**() RETURN BINARY_INTEGER;

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

FormatFontSize

Sets the font size for the current active Microsoft Word document.

Syntax

FUNCTION **FormatFontSize**(*fontsize* BINARY_INTEGER) RETURN BINARY_INTEGER;

where:

fontsize is the point of the font.

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Microsoft Excel Demo

The following sections detail how to install the Microsoft Excel demo and describe the APIs that it exposes. This demo is provided as an example of the types of solutions that can be built with Oracle and Microsoft Excel.

The Microsoft Excel demo provides a PL/SQL package (ORDEExcel) that exposes several APIs for manipulating Microsoft Excel. Also, the Microsoft Excel demo includes a script to demonstrate the capabilities of exchanging data between Oracle and Microsoft Excel. The `exceldem.sql` script exchanges data from the EMP table in Oracle to a Microsoft Excel worksheet and puts it in a graph. Run this script after installing the demo.

Installing the Microsoft Excel Demo

Microsoft Excel must be installed on the local computer before installing this demo.

To install the Microsoft Excel demo, perform the following steps:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com\demos
```

2. Start Server Manager:

```
C:\> svrmgr1
```

3. Connect to the Oracle database instance as the user that will use the Microsoft Excel demo. For example, SCOTT/TIGER.

```
SVRMGR> CONNECT SCOTT/TIGER
```

4. Run the `excelsol.sql` script at the Server Manager prompt:

```
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\DEMOS\EXCELSOL.SQL
```

This script creates the ORDEExcel package in the current user's schema. You will receive several ORA-04043: object XXXX does not exist messages when you run this script for the first time. These messages are normal.

Using the Microsoft Excel Demo

To run the Microsoft Excel demo, perform the following steps:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com\demos
```

2. Start Server Manager:

```
C:\> svrmgr1
```

3. Connect to the Oracle database instance as the user that will use the Microsoft Excel demo. For example, SCOTT/TIGER.

```
SVRMGR> CONNECT SCOTT/TIGER
```

4. Run the `exceldem.sql` script at the Server Manager prompt:

```
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\DEMOS\EXCELDDEM.SQL
```

This script creates a Microsoft Excel spreadsheet (`excelxxxxx.xls`) in the `C:\` directory. The document contains data from the EMP table.

5. Open the `excelxxxxx.xls` file, where `xxxxx` is a timestamp, to see its contents.

Core Functionality

The following subsections describe the APIs that the Microsoft Excel demo exposes. These APIs are primitive. Be aware that much of the functionality that Microsoft Excel exposes through OLE Automation is not exposed through these APIs. These APIs and PL/SQL code are provided as a “proof of concept” that Oracle COM Automation feature is viable.

CreateExcelWorkSheet

Starts the Microsoft Excel OLE Automation server and instantiates the objects for a workbook and a worksheet.

Syntax

```
FUNCTION CreateExcelWorkSheet() RETURN BINARY_INTEGER;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

InsertData

Inserts any kind of data into a specific cell of the current active Excel worksheet.

Syntax

```
FUNCTION InsertData(range VARCHAR2, data ANY PL/SQL DATATYPE, datatype VARCHAR2)  
RETURN BINARY_INTEGER;
```

where:

<i>range</i>	a string that indicates a specific cell in the current Excel worksheet (for example, 'A1', 'B1').
<i>data</i>	is the data that you want to insert into the current Excel worksheet.
<i>datatype</i>	a string that indicates the datatype of the data that you are inserting into Excel. The list of available datatypes are: <ul style="list-style-type: none">▪ I2 - 2 byte integer▪ I4 - 4 byte integer▪ R4 - IEEE 4 byte real▪ R8 - IEEE 8 byte real▪ SCODE - error code▪ CY - currency▪ DISPATCH - dispatch pointer▪ BSTR - String▪ BOOL - boolean▪ DATE - date

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

InsertChart

Creates a chart of a specified range of data and inserts the chart at the x and y position of the current worksheet with the desired height and width.

Syntax

```
FUNCTION InsertChart(xpos BINARY_INTEGER, ypos BINARY_INTEGER, width BINARY_  
INTEGER, height BINARY_INTEGER, range VARCHAR2, type VARCHAR2) RETURN BINARY_  
INTEGER;
```

where:

<i>xpos</i>	is the x position in the current worksheet where the chart should be inserted.
<i>ypos</i>	is the y position in the current worksheet where the chart should be inserted.
<i>width</i>	is the width of the chart.
<i>height</i>	is the height of the chart.
<i>range</i>	is the range of cells to be graphed.
<i>type</i>	is the datatype of the data to be graphed.

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

SaveExcelFile

Saves the current active Microsoft Excel workbook as a specific file.

Syntax

```
FUNCTION SaveExcelFile(filename VARCHAR2) RETURN BINARY_INTEGER;  
where:
```

filename is the fully qualified file name of the Excel workbook.

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

ExitExcel

Performs some cleanup and destroys the outstanding references to the Excel OLE Automation server. This should be the last API called.

Syntax

```
FUNCTION ExitExcel() RETURN BINARY_INTEGER;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Microsoft PowerPoint Demo

The following sections detail how to install the Microsoft PowerPoint demo and describe the APIs that it exposes. This demo is provided as an example of the types of solutions that can be built with Oracle and Microsoft PowerPoint.

The Microsoft PowerPoint demo provides a PL/SQL package (ORDPPT) that exposes several APIs for manipulating Microsoft PowerPoint. Also, the Microsoft PowerPoint demo includes a script (`pptdem.sql`) to demonstrate the capabilities of exchanging data between Oracle and Microsoft PowerPoint. Run this script after installing the demo.

Installing the Microsoft PowerPoint Demo

Microsoft PowerPoint97 must be installed on the local computer before installing this demo.

To install the Microsoft PowerPoint demo, perform the following steps:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com\demos
```

2. Start Server Manager:

```
C:\> svrmgrl
```

3. Connect to the Oracle database instance as the user that will use the Microsoft PowerPoint demo. For example, SCOTT/TIGER.

```
SVRMGR> CONNECT SCOTT/TIGER
```

4. Run the `pptsol.sql` script at the Server Manager prompt:

```
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\DEMOS\PPTSOL.SQL
```

This script creates the ORDPPT package in the current user's schema. You will receive several ORA-04043: object XXXX does not exist messages when you run this script for the first time. These messages are normal.

Using the Microsoft PowerPoint Demo

To run the Microsoft PowerPoint demo, perform the following steps:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com\demos
```

2. Start Server Manager:

```
C:\> svrmgr1
```

3. Connect to the Oracle database instance as the user that will use the Microsoft PowerPoint demo. For example, SCOTT/TIGER.

```
SVRMGR> CONNECT SCOTT/TIGER
```

4. Run the `pptdem.sql` script at the Server Manager prompt:

```
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\DEMOS\PPTDEM.SQL
```

This script creates a Microsoft PowerPoint presentation (`pptdemo.ppt`) on `C:\`. The document contains a list of table names owned by the user account, such as SCOTT.

5. Open `pptdemo.ppt` to see its contents.

Core Functionality

The following subsections describe the APIs that the Microsoft PowerPoint demo exposes. These APIs are primitive. Be aware that much of the functionality that Microsoft PowerPoint exposes through OLE Automation is not exposed through these APIs. These APIs and PL/SQL code are provided as a “proof of concept” that Oracle COM Automation feature is viable.

CreatePresentation

Starts the Microsoft PowerPoint OLE Automation server and instantiates the objects for a presentation.

Syntax

```
FUNCTION CreatePresentation (servername IN varchar2) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

AddSlide

Inserts a new slide in the PowerPoint presentation.

Syntax

```
FUNCTION AddSlide (layout IN binary_integer) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

SetTitle

Specifies the title of the PowerPoint slide.

Syntax

```
FUNCTION SetTitle (title IN varchar2) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

InsertText

Inserts text into the specified location on the slide.

Syntax

```
FUNCTION InsertText (orientation IN binary_integer, left IN binary_integer, top  
IN binary_integer, width IN binary_integer, height IN binary_integer, text IN  
VARCHAR2) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

PresentationSave

Save the current PowerPoint presentation.

Syntax

```
FUNCTION PresentationSave RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

PresentationSaveAs

Saves the current presentation using the specified name.

Syntax

```
FUNCTION PresentationSaveAs (filename IN varchar2) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

PresentationClose

Closes the current PowerPoint presentation.

Syntax

```
FUNCTION PresentationClose RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Exit

Exits the PowerPoint program.

Syntax

```
FUNCTION Exit RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

MAPI Demo

The following sections detail how to install the MAPI demo and describe the APIs that it exposes. This demo is provided as an example of the types of solutions that can be built with Oracle and Messaging Application Programming Interface (MAPI) compliant applications.

The MAPI demo provides a PL/SQL package (ORDMAPI) that exposes several APIs for manipulating MAPI. Also, the MAPI demo includes a script (`mapidem.sql`) to demonstrate the capabilities of exchanging data between Oracle and MAPI. Run this script after installing the demo.

Installing the MAPI Demo

The MAPI application, such as MS Outlook 98 or Oracle InterOffice, must be installed on the local computer before installing this demo.

To install the MAPI demo, perform the following steps:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com\demos
```

2. Start Server Manager:

```
C:\> svrmgrl
```

3. Connect to the Oracle database instance as the user that will use the MAPI demo. For example, SCOTT/TIGER.

```
SVRMGR> CONNECT SCOTT/TIGER
```

4. Run the `mapisol.sql` script at the Server Manager prompt:

```
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\DEMOS\MAPISOL.SQL
```

This script creates the ORDMAPI package in the current user's schema. You will receive several ORA-04043: object XXXX does not exist messages when you run this script for the first time. These messages are normal.

Using the MAPI Demo

To run the MAPI demo, perform the following steps:

1. Change to the following directory from the MS-DOS command prompt:

```
C:\> cd oracle_base\oracle_home\com\demos
```

2. Start Server Manager:

```
C:\> svrmgr1
```

3. Connect to the Oracle database instance as the user that will use the MAPI demo. For example, SCOTT/TIGER.

```
SVRMGR> CONNECT SCOTT/TIGER
```

4. Run the `mapidem.sql` script at the Server Manager prompt:

```
SVRMGR> @ORACLE_BASE\ORACLE_HOME\COM\DEMOS\MAPIDEM.SQL
```

This script connects to an Oracle database, extracts the data, and sends an email to a specified recipient.

Core Functionality

The following subsections describe the APIs that the MAPI demo exposes. These APIs are primitive. Be aware that much of the functionality that MAPI exposes through OLE Automation is not exposed through these APIs. These APIs and PL/SQL code are provided as a “proof of concept” that Oracle COM Automation feature is viable.

CreateMAPISession

Starts the MAPI OLE Automation server and instantiates the objects for a session.

Syntax

```
FUNCTION CreateMAPISession (servername IN varchar2 DEFAULT '', profilename IN  
varchar2 DEFAULT NULL, password IN varchar2 DEFAULT NULL) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

NewMessage

Creates a new message.

Syntax

```
FUNCTION NewMessage RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

AddRecipient

Adds the email address of a recipient. These are addresses where the email message will be sent.

Syntax

```
FUNCTION AddRecipient (emailaddress varchar2) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

SetSubject

Specifies the subject of the email message.

Syntax

```
FUNCTION SetSubject (subject varchar2) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

SetBody

Inserts the body text of the email message.

Syntax

```
FUNCTION SetBody (messagetext varchar2) RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

SendMessage

Sends the email message to the specified recipients.

Syntax

```
FUNCTION SendMessage RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

EndMAPISession

Exits the MAPI session.

Syntax

```
FUNCTION EndMAPISession RETURN binary_integer;
```

Remarks

This procedure returns a 0 when successful or a non-zero HRESULT when an error occurs.

Glossary

Component Object Model (COM)

A binary standard that enables objects to interact with other objects, regardless of the programming language that each object was written in.

Distributed Component Object Model (DCOM)

An extension of COM that enables objects to interact with other objects across a network.

Dynamic Link Library (DLL)

An executable file that a Windows application can load when needed.

external procedure

A function written in a third-generation language (3GL), such as C, and callable from within PL/SQL or SQL as if it were a PL/SQL function or procedure.

GUID

An identifier that uniquely identifies a COM object. GUID is an acronym for Globally Unique Identifier.

IID

A [GUID](#) that identifies a COM interface.

listener

The server process that listens for and accepts incoming connection requests from client applications. Oracle listener processes start up Oracle database processes to handle subsequent communications with the client.

listener.ora

A configuration file that describes one or more Transparent Network Substrate (TNS) listeners on a server.

Messaging Application Programming Interface (MAPI)

MAPI is a messaging architecture composed of a set of common application programming interfaces that enables multiple applications to interact with multiple messaging systems across a variety of hardware platforms.

Net8

The Oracle client/server communication software that offers transparent operation to Oracle tools or databases over any type of network protocol and operating system.

Optimal Flexible Architecture (OFA)

A set of file naming and placement guidelines for Oracle software and databases.

Oracle COM Automation feature

An Oracle feature that enables PL/SQL developers to programmatically manipulate COM objects through the OLE Automation interface (IDispatch).

PL/SQL

Oracle Corporation's procedural language extension to SQL.

progID

A descriptive string that maps to a [GUID](#).

tnsnames.ora

A file that contains connect descriptors mapped to net service names. The file may be maintained centrally or locally, for use by all or individual clients.

Index

A

- application programming interfaces
 - PL/SQL, 4-6
- architecture
 - models 1 and 2, 1-7
 - Oracle COM Automation option, 1-4

B

- benefits
 - Oracle COM Automation option, 1-3

C

- COM automation
 - errors, 4-16
- COM objects
 - program ID, 4-3
 - properties and methods, 4-3
 - required information, 4-3
 - viewing, 4-4
- com81.dll, 1-5, 2-2
- Component Object Model
 - see COM
- comwrap.sql, 2-2, 3-2
- configuring
 - DCOM, 3-5
 - listener for Oracle COM Automation, 3-3
 - Oracle COM Automation option, 3-2
- core functionality
 - Oracle COM Automation option, 4-2
- CreateObject, PL/SQL API, 4-6

D

- datatypes
 - conversion, 4-5
 - PL/SQL to Visual Basic, 4-5
- DCOM
 - configuring, 3-5
- demos
 - installing MAPI demo, 5-15
 - installing Microsoft Excel demo, 5-7
 - installing Microsoft PowerPoint demo, 5-11
 - installing Microsoft Word demo, 5-2
 - MAPI, 5-15
 - Microsoft Excel, 5-7
 - Microsoft PowerPoint, 5-11
 - Microsoft Word, 5-2
 - Oracle COM Automation, 5-2
 - program requirements for Oracle COM Automation, 2-2
- DestroyObject
 - PL/SQL API, 4-8
- directories
 - Optimal Flexible Architecture, 3-2
- Distributed Component Object Model
 - see DCOM
- documentation
 - generic, xxi

E

- errors
 - codes, 4-6
 - COM automation, 4-16
 - HRESULT, 4-6

- messages, 4-18
- OLE automation, 4-18
- examples
 - MAPI, 5-15
 - Microsoft Excel, 5-7
 - Microsoft PowerPoint, 5-11
 - Microsoft Word, 5-2
- EXTPROC process, 1-7
- extproc.exe, 1-5

G

- GetLastError
 - PL/SQL API, 4-8
- GetProperty
 - PL/SQL API, 4-10
- Globally Unique Identifier (GUID), 4-3

H

- HRESULT
 - return codes, 4-6

I

- InitArg
 - PL/SQL API, 4-12
- installation
 - Oracle COM Automation, 2-1
- Invoke
 - PL/SQL API, 4-15

L

- listener
 - configuring for Oracle COM Automation, 3-3

M

- MAPI
 - demo script mapidem.sql, 5-15
 - example, 5-15
- Messaging Application Programming Interface
 - see MAPI
- Microsoft Excel
 - demo script exceldem.sql, 5-7

- example, 5-7
- Microsoft PowerPoint
 - demo script pptdem.sql, 5-11
 - example, 5-11
- Microsoft Word
 - demo script worddem.sql, 5-2
 - example, 5-2

O

- OFA
 - Optimal Flexible Architecture, 3-2
- OLE automation
 - errors, 4-18
 - invoking, 1-5
- OLE/COM Object Viewer, 4-4
- Optimal Flexible Architecture, 3-2
- ORA-28575 error message, 3-4
- Oracle base
 - described, xx
- Oracle COM Automation option
 - architecture, 1-4
 - benefits, 1-3
 - components, 2-2
 - configuring, 3-2
 - core functionality, 4-2
 - defined, 1-2
 - demos, 5-2
 - installing, 2-1
 - introduction, 1-2
- Oracle home
 - described, xx
- ORDExcel
 - PL/SQL package, 5-7
- ORDMAP
 - PL/SQL package, 5-15
- ORDPPT
 - PL/SQL package, 5-11
- ORDWord
 - PL/SQL package, 5-2

P

- PL/SQL
 - application programming interfaces (APIs), 4-6

- ORDCOM package, 4-6
- ORDExcel package, 5-7
- ORDMAPI package, 5-15
- ORDPPT package, 5-11
- ORDWord package, 5-2
- progID
 - COM objects, 4-3
- program ID
 - COM objects, 4-3

R

- requirements
 - demo programs for Oracle COM Automation, 2-2
- return codes
 - HRESULT, 4-6

S

- SetArg
 - PL/SQL API, 4-13
- SetProperty
 - PL/SQL API, 4-11
- SetPtrArg
 - PL/SQL API, 4-14
- system requirements
 - Oracle COM Automation, 2-2

T

- troubleshooting
 - COM automation errors, 4-16
 - OLE automation errors, 4-18

