Oracle®

Developer's Guide for Microsoft Transaction Server

Release 9.0.1 for Windows

June 2001

Part No. A90170-01

ORACLE®

Oracle Developer's Guide for Microsoft Transaction Server, Release 9.0.1 for Windows

Part No. A90170-01

Authors: Mark Kennedy and Tamar Rothenberg

Contributors: Vivek Raja and Eric Wang

Contents

3 Managing Recovery Scenarios

4 Running the Microsoft Application Demo

5 Programming with Microsoft Transaction Server and an Oracle Database Server

Send Us Your Comments

Oracle Developer's Guide for Microsoft Transaction Server, Release 9.0.1 for Windows

Part No. A90170-01

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

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

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

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

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

If you have problems with the software, please contact your local Oracle Support Services. Contact information for Oracle Support Services is available at this Web site:

`http://www.oracle.com/support/`

Preface

This guide is the primary source of introduction, installation, configuration, usage, and administration information for using **Microsoft Transaction Server** with Oracle.

This preface contains these topics:

Audience

Oracle Developer's Guide for Microsoft Transaction Server is intended for application developers who perform the following tasks:

- Use **component object model (COM)** components with Microsoft Transaction Server
- Register COM components as transactional and have Microsoft Transaction Server control the transaction
- Use client-side connection pooling in Microsoft Transaction Server

Organization

This document contains:

Chapter 1, "Using Microsoft Transaction Server with Oracle"

This chapter describes the integration of Microsoft Transaction Server and an Oracle database server.

Chapter 2, "Installing and Migrating Oracle Products"

This chapter describes installation and migration requirements for Microsoft Transaction Server and database server environments.

Chapter 3, "Managing Recovery Scenarios"

This chapter describes how to schedule Microsoft Transaction Server transaction recovery.

Chapter 4, "Running the Microsoft Application Demo"

This chapter describes how to use the sample Microsoft COM-based application demo that is integrated with Microsoft Transaction Server.

Chapter 5, "Programming with Microsoft Transaction Server and an Oracle Database Server"

This chapter describes how to program with Microsoft Transaction Server and a database server.

Chapter 6, "Tuning Microsoft Transaction Server Performance"

This chapter describes how to tune Microsoft Transaction Server performance.

Chapter 7, "Troubleshooting Oracle Services for Microsoft Transaction Server"

This chapter describes how to troubleshoot Oracle Services for Microsoft Transaction Server problems.

Appendix A, "Using Oracle Services for Microsoft Transaction Server on Windows 2000"

This appendix describes differences between using Oracle Services for Microsoft Transaction Server on Windows NT and Windows 2000.

Glossary

Related Documentation

For more information, see these Oracle resources:

- The Oracle documentation set, especially:
  - Oracle9*i* Database installation guide for Windows
  - *Oracle9i Database Getting Started for Windows*
  - *Oracle9i Database Reference*
  - *Oracle Provider for OLE DB Developer's Guide*
  - *Oracle Objects for OLE*
  - *Oracle9i Net Services Administrator's Guide*

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

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

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

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

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

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

`http://technet.oracle.com/membership/index.htm`

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

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

Conventions

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

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

Conventions in Text

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

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

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

Conventions in Code Examples

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

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

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

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

Convention	Meaning	Example
. . .	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS subquery;` `SELECT col1, col2, ... , coln FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

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

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

Conventions for Windows Operating Systems

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

Convention	Meaning	Example
Choose Start >	How to start a program. For example, to start Oracle Database Configuration Assistant, you must click the Start button on the taskbar and then choose Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant.	Choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant
File and Directory Names	File / directory names are not case sensitive. The special characters <, >, :, ", /, |, and - are not allowed. The special character \ is treated as an element separator, even when it appears in quotes. If the file name begins with \ \, Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32` is the same as `C:\WINNT\SYSTEM32`
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is "^". Your prompt reflects the subdirectory in which you are working. Referred to as the command prompt in this manual.	`C:\oracle\oradata>`

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

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

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default was: ■ `C:\orant` for Windows NT ■ `C:\orawin95` for Windows 95 ■ `C:\orawin98` for Windows 98 or whatever you called Oracle home. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle`. If you install Oracle9*i* Release 1 (9.0.1) on a computer with no other Oracle software installed, the default setting for the first Oracle home directory is `C:\oracle\ora90`. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. See *Oracle9i Database Getting Started for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be

accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

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

What's New in Oracle Services for Microsoft Transaction Server?

This section describes new features of **Oracle Services for Microsoft Transaction Server** release 9.0 and provides pointers to additional information.

The following section describes the new Oracle Services for Microsoft Transaction Server features:

- Oracle9i Release 9.0 New Features in Oracle Services for Microsoft Transaction Server

Oracle9*i* Release 9.0 New Features in Oracle Services for Microsoft Transaction Server

Oracle Services for Microsoft Transaction Server releases prior to 9.0 required a Windows NT service named **Oracle Service for MTS** to be created for each Oracle database server. This enabled the database to participate in **Microsoft Transaction Server** transactions. However, only one Oracle Service for MTS was supported for each database server. The Oracle Service for MTS is no longer required. All code and logic to enable database servers to participate in Microsoft Transaction Server transactions are now embedded in the Microsoft Transaction Server application process.

Because of this, this release offers:

- Better performance

  Communication between the Microsoft Transaction Server application and the Oracle Service for MTS is no longer required.

- High availability

  The database server is no longer dependent on the Oracle Service for MTS. Previously, if the Oracle Service for MTS was stopped, the database server was unable to participate in Microsoft Transaction Server transactions.

- Improved scalability

  The code that enables a database server to participate in Microsoft Transaction Server transactions is now embedded in each Microsoft Transaction Server application process.

- Minimal configuration is necessary

  The **Oracle MTS Recovery Service** is automatically created and configured on each computer in the middle tier during Oracle Services for Microsoft Transaction Server installation.

See Also: Figure 1–1, "Microsoft Transaction Server and Database Server Integration" on page 1-4

- Application Programmatic Interface (API) changes

  Changes have also been made to the call level interface (CLI) that starts the enlistment of Oracle connections in Microsoft Transaction Server-started transactions. `OraMTSSvcEnlist()` and `OraMTSSvcEnlistEx()` are provided for backward compatibility only. In addition, there are three new APIs:

  - `OraMTSEnlCtxGet()`
  - `OraMTSEnlCtxRel()`
  - `OraMTSJoinTxn()`

  See Also: "OCI Integration with Microsoft Transaction Server Overview" on page 5-9

Note: The **Oracle Manager for MTS Services snap-in** for the **Microsoft Management Console** is no longer included. This snap-in was required to create the Oracle Service for MTS in releases prior to Oracle9*i* release 9.0.

1

Using Microsoft Transaction Server with Oracle

This chapter describes **Microsoft Transaction Server** and Oracle database server integration.

This chapter contains these topics:

Microsoft Transaction Server Overview

Microsoft Transaction Server is a proprietary **component object model (COM)** transaction processing system that runs on an Internet or network server. Microsoft Transaction Server deploys and manages application and database transaction requests on behalf of a client computer. Microsoft Transaction Server provides the following:

- An ActiveX/**distributed component object model (DCOM)** programming model to develop distributed applications and a runtime environment in which to deploy these applications
- **Atomicity, Consistency, Isolation, and Durability (ACID)** properties to components in transactions
- Access to performance enhancing features such as component caching and database connection pooling

Microsoft Transaction Server is a component of the three-tiered, server-centric architecture model. This enables the presentation, business logic, and data elements of applications to be clearly separated onto different computers connected in a network.

> **See Also:** Microsoft's documentation for additional information about Microsoft Transaction Server

Microsoft Transaction Server and Oracle Integration Overview

Without any special integration, you can deploy a COM component in Microsoft Transaction Server that connects to an Oracle database server release 8.0.6 or higher. To use either of the following features, however, you must install **Oracle Services for Microsoft Transaction Server**:

- Register the COM component as transactional (using the Properties dialog box of the component in the **Microsoft Management Console** Explorer) and have Microsoft Transaction Server control the transaction
- Use client-side connection pooling in Microsoft Transaction Server

After Oracle Services for Microsoft Transaction Server is installed, an **Oracle MTS Recovery Service** is also automatically installed on the same computer. The Oracle MTS Recovery Service coordinates Microsoft Transaction Server-related Oracle transactions originating from that computer. On each database server to which the COM component connects, you perform the following:

- Create the Microsoft Transaction Server administrator user account.
- Schedule a database server-level transaction recovery job.

This enables the database server to participate in Microsoft Transaction Server-started transactions.

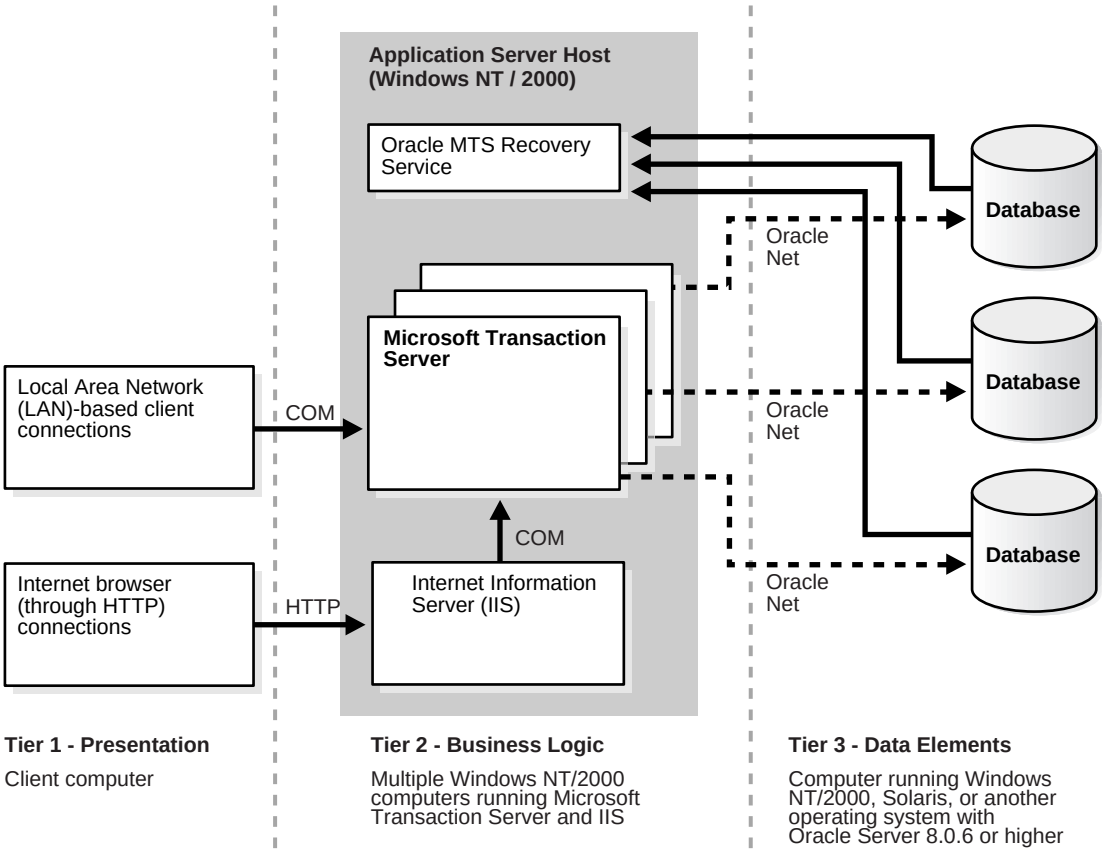
Create the COM component with any of the following Oracle products:

- **Oracle Call Interface (OCI)**
- **Oracle Objects for OLE (OO4O)**
- **Oracle Open Database Connectivity (ODBC) Driver**
- **Oracle Provider for OLE DB**

Additional application programmatic interfaces (APIs) may be integrated in the future.

Figure 1–1 shows how Microsoft Transaction Server and the database server are integrated into the three-tiered, server-centric model.

Figure 1–1 Microsoft Transaction Server and Database Server Integration



Getting Started with Microsoft Transaction Server and Oracle

You are now ready to use Microsoft Transaction Server with a database server. To get started quickly, follow the chapters identified in Table 1–1 in the order listed.

Table 1–1 Getting Started with Microsoft Transaction Server and Oracle

To Perform The Following Task...	See...
■ Install the Oracle and Microsoft products required for Microsoft Transaction Server and database server integration. ■ Migrate from a previous release of Oracle Services for Microsoft Transaction Server.	Chapter 2, "Installing and Migrating Oracle Products"
■ Create the Microsoft Transaction Server administrator user account. ■ Schedule a Microsoft Transaction Server transaction recovery job.	Chapter 3, "Managing Recovery Scenarios"
Run the Microsoft application demo.	Chapter 4, "Running the Microsoft Application Demo" for information about running an application demo that: ■ Uses transactional COM components hosted by Microsoft Transaction Server ■ Accesses a database server in a transaction controlled by Microsoft Transaction Server
Create Microsoft Transaction Server-hosted COM applications.	Chapter 5, "Programming with Microsoft Transaction Server and an Oracle Database Server" for instructions on using OCI, OO4O, Oracle ODBC Driver, or Oracle Provider for OLE DB with COM-based applications.
Learn about differences between Microsoft Transaction Server on Windows NT and Windows 2000.	Appendix A, "Using Oracle Services for Microsoft Transaction Server on Windows 2000"

2

Installing and Migrating Oracle Products

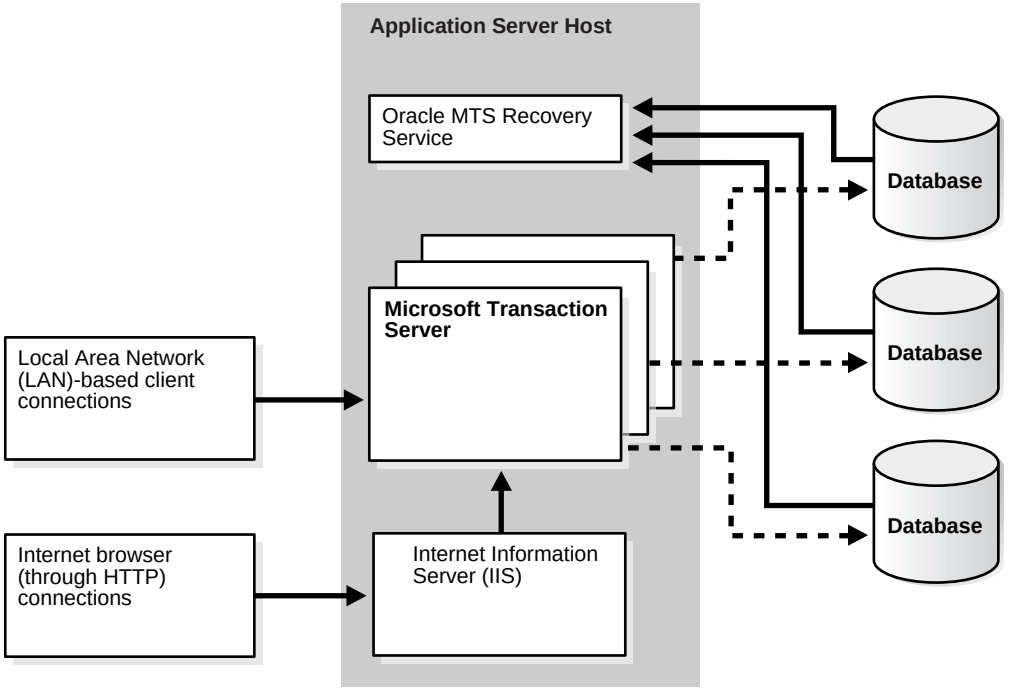
This chapter describes installation and migration requirements for the **Microsoft Transaction Server** and Oracle database server environment.

This chapter contains these topics:

Microsoft Transaction Server and Oracle Installation Overview

Figure 2–1 shows a typical Microsoft Transaction Server and multiple database server installation layout.

Figure 2–1 Microsoft Transaction Server and Multiple Database Server Installation Layout



Oracle Services for Microsoft Transaction Server Installation Requirements

Table 2–1 lists the Oracle and non-Oracle products you must install. After reviewing the installation requirements, see the Oracle9*i* Database installation guide for Windows for instructions on installing the required Oracle products. Key guidelines to understand are:

- At least 256 MB of RAM is required if the database server, Microsoft Transaction Server, and **Oracle Services for Microsoft Transaction Server** are installed on the same computer.
- An **Oracle MTS Recovery Service** is also automatically installed on the same computer in the middle tier as Oracle Services for Microsoft Transaction Server. One Oracle MTS Recovery Service per computer is allowed.
- For **Oracle Fail Safe** cluster configurations (or any failover configuration), install Microsoft Transaction Server on the node running the **Microsoft Distributed Transaction Coordinator (MS DTC)** component. This ensures that the Oracle MTS Recovery Service migrates with the client application during failover. You can configure this when scheduling recovery transactions.

Note: Microsoft Transaction Server integration with a database server is not currently supported on Windows 95 or Windows 98.

Table 2–1 Installation Requirements

	What Products Must Be Installed?	
On This Computer...	**Oracle Products**	**Non-Oracle Products**
Client computer	■ None required	■ Fully-functioning networking protocol software ■ Web browser
Windows NT computer where Microsoft Transaction Server is installed	■ Oracle Services for Microsoft Transaction Server[1] ■ Oracle Net Services for the client[2] ■ Oracle Objects for OLE (OO4O)[3] ■ Oracle Open Database Connectivity (ODBC) Driver[3] ■ Oracle Provider for OLE DB[3] ■ Oracle Call Interface (OCI)[3] ■ Oracle Net Manager ■ SQL*Plus	■ Microsoft Internet Information Server (IIS)[4] ■ Microsoft Transaction Server release 2.0[5] ■ Windows NT 4.0 Service Pack 5 or higher[6] ■ Fully-functioning networking protocol software
Computer where the database server is installed	■ Oracle Server (the database server)[7] ■ Oracle Net Services for the server ■ SQL*Plus	■ Fully-functioning networking protocol software

[1] See the Oracle9*i* Database installation guide for Windows for installation instructions. During installation, you are prompted to enter the port number on which the Oracle MTS Recovery Service will listen for requests to resolve in-doubt transactions.

[2] Oracle Net Services is automatically installed with Oracle Services for Microsoft Transaction Server.

[3] The 9.0 releases of OO4O, Oracle ODBC Driver, Oracle Provider for OLE DB, and OCI are only required if you are building or using components with which they are required.

[4] An applications server like IIS is not required, but is beneficial if using Microsoft Transaction Server. Install the applications server on the same computer as Microsoft Transaction Server. The applications server you choose must support extensions.

[5] Microsoft Transaction Server is currently available as part of the Microsoft Windows NT 4.0 Option Pack. **Microsoft Management Console** is automatically installed with Microsoft Transaction Server.

[6] After installing Microsoft Transaction Server 2.0, Oracle Corporation recommends that you install Windows NT 4.0 Service Pack 5 or higher. You must re-install any service packs installed prior to Microsoft Transaction Server 2.0 installation.

[7] If the database server is a release prior to 8.0.6, you cannot use the integration described in this guide. See "Using Microsoft's Oracle ODBC Driver" on page 5-28 for a description of what functionality is available for pre-8.0.6 database servers.

Migrating From a Previous Oracle Services for Microsoft Transaction Server Installation

Starting with Oracle9*i* release 9.0, you no longer have to create the **Oracle Service for MTS**. However, before proceeding to uninstall Oracle Services for Microsoft Transaction Server, you must use the **Oracle Manager for MTS Services snap-in** in the Microsoft Management Console Explorer to delete the existing Oracle Service for MTS. Table 2–2 shows the procedures to follow.

Table 2–2 Migration Requirements

If You...	Then...	See...
Migrate the release 8.1.*x* database server to the current release and create a new installation of release 9.0	**1.** Delete the Oracle Service for MTS using the Oracle Manager for MTS Services snap-in. **Note:** If you have already deleted the database server, you cannot delete the Oracle Service for MTS using the Oracle Manager for MTS Services snap-in. Instead, delete the service from the registry. This is not the preferred way.	"Deleting an Oracle Service for MTS with the Oracle Manager for MTS Services Snap-In" on page 2-6 "Using the Registry to Manually Delete the Oracle Service for MTS" on page 2-11 if you have already deleted the database server
	2. Delete roles and privileges of the user associated with the deleted Oracle Service for MTS.	"Deleting Roles and Privileges of an Inactive Oracle Service for MTS User" on page 2-10
	3. Uninstall Oracle Services for Microsoft Transaction Server from the Windows NT computer where Microsoft Transaction Server is installed. **4.** Install the 9.0 release of OO4O, Oracle Provider for OLE DB, Oracle ODBC Driver, or OCI, if you plan to build component object model (COM) components with these products. **5.** Install Oracle Services for Microsoft Transaction Server release 9.0 into a single Oracle home. The Oracle MTS Recovery Service is also automatically installed.	The Oracle9*i* Database installation guide for Windows
	6. Create the Microsoft Transaction Server administrator user account **7.** Schedule Microsoft Transaction Server transaction recovery jobs for all database servers that participate in Microsoft Transaction Server transactions.	Chapter 3, "Managing Recovery Scenarios"

Deleting an Oracle Service for MTS with the Oracle Manager for MTS Services Snap-In

You must use the Oracle Manager for MTS Services snap-in in the Microsoft Management Console Explorer to delete the Oracle Service for MTS. Deleting the Oracle Service for MTS in any other way (such as with the keyboard's Delete button) causes data inconsistencies in the database server. These inconsistencies require the database administrator to manually commit or abort transactions that did not successfully complete or recover. Before deleting the Oracle Service for MTS, ensure that all transactions are resolved by performing the following tasks.

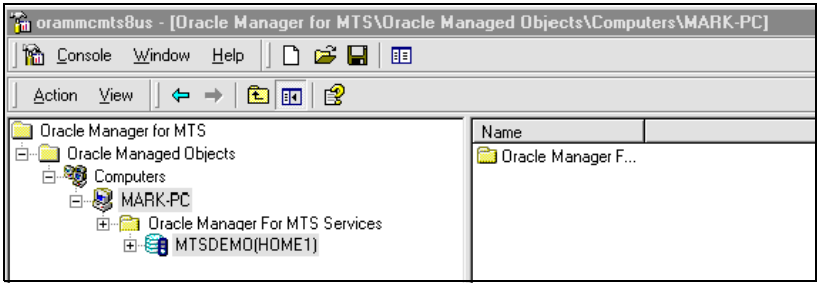
Task 1: Stop the Oracle Service for MTS

To stop the Oracle Service for MTS:

1. Go to the computer from which to delete an Oracle Service for MTS. You must modify an Oracle Service for MTS before deleting it. The Oracle Service for MTS can be running on this computer or on a remote computer that you can access from this computer.
2. Choose Start > Programs > Oracle - *HOME_NAME* > Application Development > Oracle Manager for Microsoft Transaction Server.

   The Microsoft Management Console appears.
3. Find the Oracle Service for MTS to modify in the Explorer window.

4. Right-click the Oracle Service for MTS icon to modify (named MTSDEMO in this example):



   A menu appears with several options.

5. Choose Stop Service.

   A message indicates that the Oracle Service for MTS has stopped.

6. Click OK.

Task 2: Stop and Restart the Database Server

To stop and restart the database server:

1. Go to the computer on which the database server is running.
2. Start SQL*Plus:

   ```
   C:\> sqlplus /NOLOG
   ```

3. Connect to the database as SYSDBA:

   ```
   ENTER USER-NAME: / AS SYSDBA
   ```

4. Shut down the database server:

   ```
   SQL> SHUTDOWN
   ```

5. Restart the database server:

   ```
   SQL> STARTUP
   ```

6. Exit SQL*Plus:

   ```
   SQL> EXIT
   ```

Task 3: Restart the Oracle Service for MTS

To restart the Oracle Service for MTS:

1. Return to the computer from which to modify the Oracle Service for MTS.
2. Choose Start > Programs > Oracle - *HOME_NAME* > Application Development > Oracle Manager for Microsoft Transaction Server.

   The Microsoft Management Console appears.
3. Find the Oracle Service for MTS to start in the Explorer window.
4. Right-click the Oracle Service for MTS icon.

   A menu appears with several options.
5. Choose Start Service.

   A message indicates that the Oracle Service for MTS started.
6. Click OK.

Task 4: Monitor the Oracle Service for MTS Trace Files

To monitor the Oracle Service for MTS trace files:

1. Do not enable any new transactions to use the Oracle Service for MTS.
2. Monitor the Oracle Service for MTS trace file for a message indicating that recovery completed successfully:

   ```
   2515156: [2096] OracleMTSService - Accepting new enlistment requests.
   ```

   This file is located in *ORACLE_BASE**ORACLE_HOME*\oramts\trace.
3. Right-click the Oracle Service for MTS icon in the Microsoft Management Console once this message appears.
4. Choose Stop Service.

   A message indicates that the Oracle Service for MTS stopped.
5. Click OK.

Task 5: Delete Oracle Service for MTS Table Information

To delete Oracle Service for MTS table information:

1. Go to the computer on which the database server is running.
2. Start SQL*Plus:

   ```
   C:\> sqlplus/NOLOG
   ```

3. Connect to the database as SYSDBA:

   ```
   ENTER USER-NAME: / AS SYSDBA
   ```

4. Delete this information from the following table:

   ```
   SQL> DROP TABLE mtsadmin_username.mts_proxy_info;
   ```

   where *mtsadmin_username* is the Oracle Service for MTS user (for example, mtssys).

   ```
   SQL> COMMIT;
   ```

Task 6: Delete the Oracle Service for MTS

To delete the Oracle Service for MTS:

1. Go to the computer from which to delete the Oracle Service for MTS. The Oracle Service for MTS can be running on this computer or on a remote computer that you can access from this computer.
2. Choose Start > Programs > Oracle - *HOME_NAME* > Application Development > Oracle Manager for Microsoft Transaction Server.

   The Microsoft Management Console appears.
3. Find the Oracle Service for MTS to delete in the Explorer window.
4. Right-click the Oracle Service for MTS icon.

   A menu appears with several options.
5. Choose Delete.
6. Go to the section listed in the following table based on the message that you receive:

If a Message Indicates That...	Go To...
The Oracle Service for MTS was successfully deleted.	"Deleting Roles and Privileges of an Inactive Oracle Service for MTS User" on page 2-10
The Oracle Service for MTS was not deleted.	"Using the Registry to Manually Delete the Oracle Service for MTS" on page 2-11

Deleting Roles and Privileges of an Inactive Oracle Service for MTS User

Ensure that you delete the roles and privileges assigned to an Oracle Service for MTS user that you no longer use or whose service you have deleted.

To delete roles and privileges of an inactive Oracle Service for MTS user:

1. Go to *ORACLE_BASE**ORACLE_HOME*\oramts\admin.
2. Open the file revokeuser.sql with a text editor.
3. Replace mts_user with the username from which to revoke roles and privileges.

   Note: This script uses the username mtssys and the password mtssys. If you have changed the password or are using an Oracle Service for MTS username other than mtssys, you must substitute the correct username and password.

4. Save the changes and exit revokeuser.sql.
5. Start SQL*Plus:

   ```
   C:\> sqlplus/NOLOG
   ```

6. Connect to the database as SYSDBA:

   ```
   ENTER USER-NAME: / AS SYSDBA
   ```

7. Run the modified script:

   ```
   SQL> @ORACLE_BASE\ORACLE_HOME\oramts\admin\revokeuser.sql;
   ```

   The roles and privileges for the user are deleted.

8. Exit SQL*Plus:

   ```
   SQL> EXIT
   ```

9. See the Oracle9*i* Database installation guide for Windows for instructions on installing the latest Oracle Services for Microsoft Transaction Server release.

Using the Registry to Manually Delete the Oracle Service for MTS

Before deleting the Oracle Service for MTS, it must be cleanly disassociated from the database server to which it connects. Sometimes this disassociation fails. Follow the instructions in this section only if:

- The deletion procedures in "Deleting an Oracle Service for MTS with the Oracle Manager for MTS Services Snap-In" on page 2-6 were unsuccessful
- You have already deleted the database server and cannot use the Oracle Manager for MTS Services snap-in

Table 2–3 describes the scenarios in which the Oracle Manager for MTS Services snap-in of the Microsoft Management Console Explorer can fail to delete or modify the Oracle Service for MTS.

Table 2–3 Oracle Service for MTS Deletion or Modification Failures

Scenario	Solution
The Oracle Manager for MTS Services snap-in cannot connect to the database server using the information in the registry.	Ensure that the database server and its listener are started. Use SQL*Plus or a different tool to verify that the database server accepts new connections.
The information in the database server does not match the information in the registry.	The Oracle Manager for MTS Services snap-in is connecting to a different database server than the one to which the Oracle Service for MTS connects. If the Oracle Manager for MTS Services snap-in and the Oracle Service for MTS run on the same computer, they may be using `tnsnames.ora` files from different Oracle homes. If they run on different computers (for example, the Oracle Manager for MTS Services snap-in is configuring a service on a remote computer), the entry in their `tnsnames.ora` file is pointing to different databases. Whether it is a local or remote problem, resolve it by ensuring that the entry in the `tnsnames.ora` file for both the Oracle Manager for MTS Services snap-in and the Oracle Service for MTS points to the same database instance.
The Oracle Manager for MTS Services snap-in cannot delete the service information stored in the database server.	The database server is unstable or is not working properly. Check if any database trace files are being created that indicate a database process failure. Trace files are located in *ORACLE_BASE*`\`*ORACLE_HOME*`\oramts\trace.`

Task 1: Manually Delete the Oracle Service for MTS with the Registry

To manually delete the Oracle Service for MTS with the registry:

1. Start the registry from the command prompt:

   ```
   C:\> regedt32
   ```

   The Registry Editor window appears.

2. Go to `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\OracleMTSService`*n*.

   where *n* is the number of the Oracle Service for MTS.

   The right-hand side of the window shows various parameters and values associated with `OracleMTSService`*n*, including those listed in the following table:

Parameter	This Parameter Contains...
ORAMTS_SUNAME	The Oracle Service for MTS username
ORAMTS_SUPWD	The password for the Oracle Service for MTS username (encrypted in the registry)
ORAMTS_ORADB	The net service name for the Oracle Service for MTS to use in connecting to the database server

3. Start SQL*Plus:

   ```
   C:\> sqlplus /NOLOG
   ```

4. Connect to the database server with the same username and net service name with which the Oracle Service for MTS connects:

   ```
   ENTER USER-NAME: username/password@net_service_name
   ```

   where *net_service_name* is the net service name for connecting to the database. The password is stored in the registry in encrypted form. Use plain text passwords when connecting with SQL*Plus.

5. Verify that the database server is the same one to which the Oracle Service for MTS connects by checking the following database information:

   ```
   SQL> SELECT NAME, DBID FROM V$DATABASE;
   ```

   which displays information similar to the following:

   ```
   NAME      DBID
   ---------------------
   ORCL      12345678
   ```

6. Check that these values match the registry values ORAMTS_DBNAME (ORCL in this example) and ORAMTS_DBID (12345678 in this example).
7. Check the service information:

   ```
   SQL> SELECT rmguid FROM mts_proxy_info;
   ```

   which displays information similar to the following:

   ```
   RMGUID

   ------------------------

   2320b23e93e09fff02a231974
   ```

8. Check that this information matches the registry value ORAMTS_RMGUID.
9. Proceed only if all values match.

   If not all values match, the database server is not the same one to which the Oracle Service for MTS connects. If you continue, Oracle Service for MTS installation on the database server fails. This can leave the database server in an inconsistent state that requires database administrator intervention to correct. The reason SQL*Plus connected to a different database server than the Oracle Service for MTS is mismatching tnsnames.ora files.
10. Delete the service information stored in the database:

    ```
    SQL> DELETE FROM mts_proxy_info;

    SQL> COMMIT;
    ```

11. Exit from SQL*Plus.

    ```
    SQL> EXIT
    ```

Task 2: Delete the OracleMTSService*n* Service

To delete the OracleMTSService*n* service:

1. Reboot the computer.
2. Choose Start > Programs > Oracle - *HOME_NAME* > Application Development > Oracle Manager for Microsoft Transaction Server.

   The Microsoft Management Console appears.
3. Find the Oracle Service for MTS to delete in the Explorer window.
4. Right-click the Oracle Service for MTS.

   A menu appears with several options.

5. Choose Delete.

   If successful, a message indicates that the Oracle Service for MTS was deleted.

6. If unsuccessful, a message indicates that the Oracle Service for MTS was not deleted. In this case, use the registry to delete the service's registry entry. Delete the following key:

   `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\OracleMTSService`*n*.

   where *n* is the number of the Oracle Service for MTS.

7. Go to "Deleting Roles and Privileges of an Inactive Oracle Service for MTS User" on page 2-10.

3

Managing Recovery Scenarios

This chapter describes how to create and schedule **Microsoft Transaction Server**-related Oracle transaction recovery.

This chapter contains these topics:

Microsoft Transaction Server Configuration Requirements

You must configure the Microsoft Transaction Server and Oracle database server environments after installing or migrating **Oracle Services for Microsoft Transaction Server**. Review Table 3–1 to identify what you must configure.

Table 3–1 Configuration Requirements

On This Computer...	Is Any Oracle Configuration Required?	See...
Client computer	No	Not applicable
Windows NT computer where Microsoft Transaction Server is installed	No	Not applicable
Computer where the database server is installed	Yes. Perform the following procedures: **1.** Run the `oramtsadmin.sql` script against the database server to: Create the Microsoft Transaction Server administrative user account (the default username is `mtssys`). Schedule automatic transaction recovery. **2.** For Oracle Fail Safe configurations, you can modify the registry values before or after running the `oramtsadmin.sql` script.	"Microsoft Transaction Server Transaction Recovery Overview" on page 3-3 "Scheduling Automatic Microsoft Transaction Server Transaction Recovery" on page 3-4 "Modifying Registry Values for Oracle Fail Safe Configurations" on page 3-11 if using Oracle Fail Safe

See Also: Non-Oracle product documentation for any configuration procedures required by those products (for example, Microsoft Internet Information Server)

Microsoft Transaction Server Transaction Recovery Overview

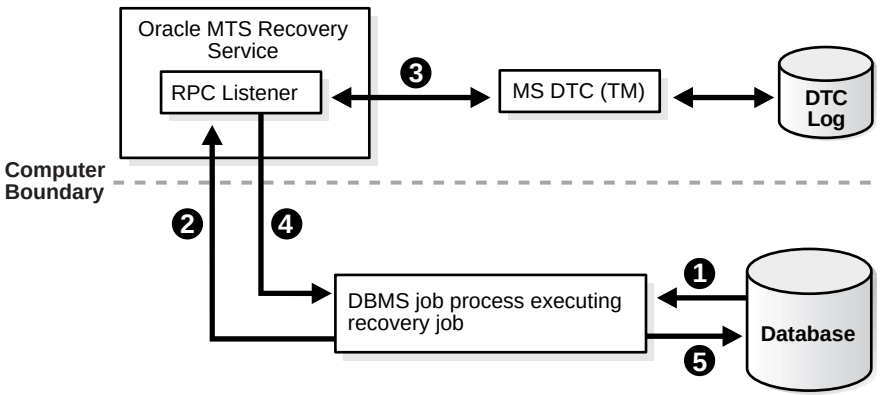
Distributed transaction capabilities are required to use Microsoft Transaction Server with Oracle. Microsoft Transaction Server-related Oracle transactions become in-doubt transactions when any of the following fail:

- Microsoft Transaction Server application
- Network
- **Microsoft Distributed Transaction Coordinator (MS DTC)**

An **Oracle MTS Recovery Service** resolves in-doubt transactions on the computer that started the failed transaction. An Oracle MTS Recovery Service is automatically installed with Oracle Services for Microsoft Transaction Server. Only one Oracle MTS Recovery Service can be installed for each computer. A scheduled recovery job on each Microsoft Transaction Server-enabled database server permits the Oracle MTS Recovery Service to resolve in-doubt transactions.

Figure 3–1 describes how in-doubt transactions are resolved.

Figure 3–1 Resolution of In-Doubt Microsoft Transaction Server Transactions



Legend
1 DBMS recovery job detects an in-doubt MTS-related transaction.
2 DBMS recovery job extracts the recovery service's endpoint address from the XID of the in-doubt transaction and requests the recovery service for the outcome of the MTS/MS DTC transaction.
3 Recovery service requests its MS DTC for transaction outcome.
4 Recovery service reports transaction outcome to DBMS job process.
5 DBMS recovery job commits or aborts in-doubt transaction.

Scheduling Automatic Microsoft Transaction Server Transaction Recovery

Automatic transaction recovery is performed by scheduling a database job. A database job for in-doubt transactions must be scheduled for each database server participating in Microsoft Transaction Server transactions.

Transaction recovery is configured by running the `oramtsadmin.sql` script. This script runs the scripts `utl_oramts.sql` and `prvtoramts.plb` to create the PL/SQL package `utl_oramts`. The database view `oramts_2pc_pending` is also created to show in-doubt transactions related to Microsoft Transaction Server transactions.

The `oramtsadmin.sql` script does the following:

- Creates the Microsoft Transaction Server administrator user account
- Automatically schedules database jobs for transaction recovery every one minute

  When the database job is run, it checks for unresolved global transactions in the database server that are related to Microsoft Transaction Server. Information in the **transaction identifiers (XIDs)** of the in-doubt transactions identifies the computer on which the transaction was started. The Oracle MTS Recovery Service on that computer resolves the transaction.
- Schedules post-recovery cleanup every half hour

Schedule automatic transaction recovery in the database server by performing the following tasks:

- Task 1: Set and Start Up SNP Processes
- Task 2: Create and Schedule Automatic Transaction Recovery

Task 1: Set and Start Up SNP Processes

The JOB_QUEUE_PROCESSES initialization parameter specifies the number of job queue (SNP) processes started in an instance.

To set and start up SNP processes:

1. Ensure that you have SYSDBA privileges.
2. Go to the computer on which the database server is installed.
3. Start SQL*Plus:

   ```
   C:\> sqlplus /NOLOG
   ```

4. Connect to the database server as SYSDBA:

   ```
   ENTER USER-NAME: / AS SYSDBA
   ```

5. Review the following table for the location of the initialization parameter file for the database server release:

If Using This Server...	Go to...
8.1.*x* or later database server	*ORACLE_BASE*\admin*DB_NAME*\pfile\init.ora
8.0.6 database server	*ORACLE_HOME*\database\init*sid*.ora

6. Open the initialization parameter file.
7. Set the following initialization parameter to at least this value:

   ```
   JOB_QUEUE_PROCESSES=1
   ```

   The default value for this parameter is 0. Set this parameter to a value greater than 1 if there are many destinations to which to propagate the messages.

8. Shut down the database server:

   ```
   SQL> SHUTDOWN
   ```

9. Restart the database server:

   ```
   SQL> STARTUP
   ```

10. Exit SQL*Plus:

    ```
    SQL> EXIT
    ```

Task 2: Create and Schedule Automatic Transaction Recovery

The `oramtsadmin.sql` script creates the Microsoft Transaction Server administrator user account with the default username `mtssys`. The Microsoft Transaction Server transaction recovery jobs run under the administrator user account.

The `oramtsadmin.sql` script runs the `utl_oramts.sql` script to grant the following privileges and roles to the administrator user account:

- `CONNECT` role
- `SELECT_CATALOG_ROLE` role
- `FORCE_ANY_TRANSACTION` privilege
- `DBMS_JOBS` package, on which `EXECUTE` privileges are granted
- `DBMS_TRANSACTION` package, on which `EXECUTE` privileges are granted

To create and schedule automatic transaction recovery:

1. Ensure that you have `SYSDBA` privileges.
2. Log on to the computer where the database server is installed.
3. Start SQL*Plus:

   ```
   C:\> sqlplus/NOLOG
   ```

4. Connect to the database server as `SYSDBA`:

   ```
   ENTER USER-NAME: / AS SYSDBA
   ```

5. Run the `oramtsadmin.sql` script:

   ```
   SQL> @ORACLE_BASE\ORACLE_HOME\oramts\admin\oramtsadmin.sql;
   ```

   You are prompted individually for the Microsoft Transaction Server administrator username and password. You can accept the default username of `mtssys` and password of `mtssys`, or change them.

6. If you do not change the password in step 5, change it for the `mtssys` user afterward:

   ```
   SQL> ALTER USER mtssys IDENTIFIED BY new_password;
   ```

> **Note:** To change the username later, drop the user, rerun the `oramtsadmin.sql` script, and specify a different username when prompted.

7. Exit SQL*Plus:

```
SQL> EXIT
```

A single PL/SQL package, `utl_oramts`, is created in the Microsoft Transaction Server administrator's schema. `utl_oramts` exposes these public procedures and creates this view:

- `utl_oramts.show_indoubt` procedure
- `utl_oramts.recover_automatic` procedure
- `utl_oramts.forget_RMs` procedure
- `oramts_2pc_pending` view

utl_oramts.show_indoubt Procedure

Type

Procedure

Arguments

None

Returns

None

Description

Use this procedure to view Microsoft Transaction Server in-doubt transactions in the database server. This procedure uses the `dbms_output` package to display results.

This procedure requires `SERVEROUTPUT` set to `ON`.

```
SQL> SET SERVEROUTPUT ON

SQL> EXECUTE utl_oramts.show_indoubt;
```

The following information appears:

```
=================================================
currently indoubt transactions
=================================================
formatid   : 21255235
gtrid      : C2229A505904974D81FB7316B147325900000000
bqual      : 5BAB6A6B55CD294AA20335839110829C0100000000901944700050
local txid : 142.11.202
tx state   : prepared
protocol   : HTTP
endpoint   : middletier-1@foo.com:2030
=================================================
formatid   : 21255235
gtrid      : 259DF9C8DFC5574F8876F0DF4E15CCAD00000000
bqual      : 2C8DCED5B9816244BA2B73CC013EEB870100000000901944700050
local txid : 2.18.185
tx state   : prepared
protocol   : HTTP
endpoint   : middletier-2@foo.com:2030
```

utl_oramts.recover_automatic Procedure

Type

Procedure

Arguments

None

Returns

None

Description

This procedure is run by the transaction recovery job.

An automatic database job is scheduled for `utl_oramts.recover_automatic`. When the job is run, it checks for unresolved global transactions in the database server that are related to Microsoft Transaction Server. Information in the XIDs of the in-doubt transactions identifies the computer on which the transaction started. The Oracle MTS Recovery Service is contacted and resolves the transactions.

utl_oramts.forget_RMs Procedure

Type

Procedure

Arguments

None

Returns

None

Description

Use this procedure to request the transaction monitor (MS DTC) to forget resolved transactions. This procedure is run by the post-recovery cleanup job.

oramts_2pc_pending View

The view `oramts_2pc_pending` is created by executing `oramtsadmin.sql`. `oramts_2pc_pending` shows in-doubt transactions in the database server. This view consists of the following columns:

Formatid

This is the `formatid` of the global transaction in the database server.

global_transaction_id

This is the transaction identifier of the Oracle global transaction corresponding to the Microsoft Transaction Server transaction. In fact, this is the globally unique identifier (GUID) of the Microsoft Transaction Server transaction.

branch_id

This shows the branch identifier of the Oracle transaction. A single Microsoft Transaction Server transaction can have multiple Oracle global transactions. This depends on the number of Microsoft Transaction Server / COM+ components that span the same Microsoft Transaction Server transaction. All these transactions have the small global transaction identifier, but different branch identifiers.

local_tx_id

A local Oracle transaction corresponds to each Microsoft Transaction Server transaction. This column shows the identifier corresponding to this local transaction.

state

This shows the state of the transaction: pending, heuristically committed, heuristically aborted, and so on.

protocol

This is the protocol that the transaction recovery job in the database server uses to communicate with the Oracle MTS Recovery Service.

endpoint

This is the endpoint of the Windows NT/2000 computer on which the Microsoft Transaction Server transaction originated. For HTTP connections, this translates to a hostname and port number.

Viewing Microsoft Transaction Server In-Doubt Transactions

To view Microsoft Transaction Server–related in-doubt transactions in the database server, use SQL*Plus to query the view `oramts_2pc_pending`.

To view Microsoft Transaction Server–related in-doubt transactions:

1. Start SQL*Plus with the Microsoft Transaction Server administrator user account:

   ```
   C:\> sqlplus mtsadmin_user/mtsadmin_password
   ```

2. Enter the following command:

   ```
   SQL> SELECT * FROM oramts_2pc_pending;
   ```

   This displays the computer on which the in-doubt transaction originated.

Modifying Registry Values for Oracle Fail Safe Configurations

In typical configurations, the MS DTC and Oracle MTS Recovery Service run on the same computer. In this way, the required information for transaction recovery is available to the Oracle-Microsoft Transaction Server integration layer.

In configurations where the Microsoft Transaction Server application is part of a Windows NT cluster (for example, the application can fail over to another node or host in the cluster), the MS DTC runs as a cluster wide resource. All cluster nodes use a single instance of the MS DTC running on any cluster node. Using **Oracle Fail Safe**, the Oracle MTS Recovery Service can also be made into a cluster wide resource.

If you have an Oracle Fail Safe configuration, make sure the following registry information is replicated on all nodes in the cluster participating in Microsoft Transaction Server transactions:

To modify registry values for Oracle Fail Safe configurations:

1. Go to the computer on which the MS DTC and Oracle MTS Recovery Service are installed.
2. Start the registry from the command prompt:

   ```
   C:\> regedt32
   ```

   The Registry Editor window appears.
3. Go to `HKEY_LOCAL_MACHINE\Software\Oracle\OracleMTSRecoveryService`.
4. Copy the registry information appearing here to all nodes in the cluster.
5. Reboot the computer on which you added the key.

4

Running the Microsoft Application Demo

This chapter describes how to use the sample Microsoft **component object model (COM)** application demo that is integrated with **Microsoft Transaction Server**.

This chapter contains these topics:

- Configuring OCI with the Microsoft Application Demo
- Configuring Oracle ODBC Driver with the Microsoft Application Demo
- Configuring Oracle Provider for OLE DB with the Microsoft Application Demo

---

Note: The **Microsoft application demo** is not included with Microsoft Transaction Server on Windows 2000.

Configuring OCI with the Microsoft Application Demo

You can use the **Oracle Call Interface (OCI)** with the sample banking application demo that Microsoft provides with Microsoft Transaction Server. In most cases, OCI is automatically integrated with the Microsoft application demo. Review Table 4–1 to determine if OCI and the Microsoft application demo are integrated, and what you can do if they have not been integrated.

Table 4–1 Verifying OCI and Microsoft Application Demo Integration

If Microsoft Transaction Server...	Then...
Is *already* installed when you install Oracle Services for Microsoft Transaction Server	Oracle Universal Installer automatically backs up and substitutes several Visual C++ files of the banking demo with files that integrate the `oci.dll` and `oramts.dll` files. This enables you to use OCI with the banking demo.
Is *not* installed when you install Oracle Services for Microsoft Transaction Server	Perform the following procedures: 1. Install Microsoft Transaction Server. 2. Back up the *ROOTDRIVE*`:\program files\mts\samples\packages\vcacct.dll` file on the Windows NT computer to a different location. 3. Copy the `vcacct.dll` file in *ORACLE_BASE*`\`*ORACLE_HOME*`\oramts\samples\account.vc\release` to the Windows NT directory listed in Step 2.

Microsoft Application Demo Overview

The Microsoft application demo is installed in the *ORACLE_BASE**ORACLE_HOME*\oramts\samples\account.vc directory and is an OCI implementation of the Visual C++ Sample Bank package that ships with Microsoft Transaction Server. The demo component uses the user account scott and password tiger to connect to a database whose **net service name** is mtsdemo. You can change this information in the oramisc.h file. The demo also uses two tables:

- account
- receipt

These tables are part of the user scott schema in the default Oracle database server created during installation. See "Ensuring the Database Includes the Proper Microsoft Application Demo Tables" for more information.

Ensuring the Database Includes the Proper Microsoft Application Demo Tables

If the default database server is not being used or the database server does not include the user scott, create the tables required to run the sample banking Microsoft application demo in the relevant user's schema.

To ensure the database server includes the proper tables:

1. Review the following table to determine if the database server includes the proper tables:

If You Create the Database Server Through These Methods...	Then...
■ Through the General Purpose, Transaction Processing, or Data Warehouse database configuration type of the Enterprise Edition, Standard Edition, or Personal Edition installation type of Oracle Universal Installer ■ Manually using a custom SQL script, and then explicitly run the scott.sql and omtssamp.sql scripts in that order against the database[1] ■ Through the following Oracle Database Configuration Assistant database configuration types: General Purpose Transaction Processing Data Warehouse	The database server includes the proper tables. Go to "Running the Microsoft Application Demo" on page 4-5.

If You Create the Database Server Through These Methods...	Then...
■ Manually using a custom SQL script, and do *not* run `omtssamp.sql` against the database ■ Using the CREATE DATABASE syntax in SQL*Plus line mode ■ In any available method on Solaris or any other operating system	The database server does *not* include the proper tables. Perform steps 1 through 6 in this section before proceeding to "Running the Microsoft Application Demo" on page 4-5.

[1] If you run `omtssamp.sql` before running `scott.sql` or do not run `scott.sql` at all, you receive numerous error messages. Ignore these messages. The product functions properly, but the sample application demo described in this chapter does not work.

If the database server does not include the proper tables, you must create them manually.

To manually create the proper tables:

1. Start SQL*Plus:

   ```
   C:\> sqlplus/NOLOG
   ```

2. Connect to the database server:

   ```
   ENTER USER-NAME: username/password@net_service_name
   ```

   where *net_service_name* is the net service name that connects to the database server. If you connect to the database server with a username other than `scott`, you must change the username and password in `oramisc.h` and rebuild `vcacct.dll`.

3. Create the user `mtsdemousr`:

   ```
   SQL> CREATE USER mtsdemousr IDENTIFIED BY mtsdemousr;
   ```

   This creates the user `mtsdemousr`, which runs the Microsoft sample application.

4. Assign the following roles to user `mtsdemousr`:

   ```
   SQL> GRANT CONNECT, RESOURCE TO mtsdemousr;
   ```

5. Connect with user `mtsdemousr`:

   ```
   SQL> CONNECT mtsdemousr/mtsdemousr@net_service_name
   ```

6. Run the following SQL script:

   ```
   SQL> @\ORACLE_BASE\ORACLE_HOME\oramts\samples\sql\omtssamp.sql
   ```

   This creates the `account` and `receipt` tables in the schema of user `mtsdemousr`.

Running the Microsoft Application Demo

This section describes how to run the Microsoft application demo.

To run the Microsoft application demo:

Note: The sample project was created using Visual C++ 5.0. To build the project using Visual C++ 6.0, the Microsoft `.idl` file must be processed with the new release of the MIDL compiler included with Visual C++ 6.0:

1. Go to the directory *ORACLE_BASE*`\`*ORACLE_HOME*`\oramts\samples\account.vc`.
2. Enter the following at the command prompt:

   ```
   midl account.idl
   ```

When opening a Visual C++ 5.0 project, Visual C++ 6.0 automatically updates it to the new release.

---

1. Ensure that you have installed the Sample Bank Client application shipped with Microsoft Transaction Server. The sample component DLLs are typically installed under *ROOTDRIVE*`:\program files\mts\samples\packages`. You can also verify installation through the Control Panel:
   a. Choose Start > Settings > Control Panel.
   b. Double-click Add/Remove Programs.

      The Add/Remove Programs Properties dialog box appears.

   c. Select Windows NT 4.0 Option Pack and click Add/Remove.

      The setup takes several seconds to initialize. When complete, the Microsoft Windows NT 4.0 Option Pack Setup dialog box appears.

   d. Click Next.

e. Click Add/Remove.

f. Scroll through the list until you find Transaction Server.

g. Double-click Transaction Server.

h. Ensure that the Transaction Server Development subcomponent check box is selected; this indicates that the demo is installed. If this subcomponent is not selected, then select it and follow Steps 1i through 1m. Otherwise, go to Step 2.

i. Click OK.

j. Click Next on the Microsoft Windows NT 4.0 Option Pack Setup dialog box.

Installation takes several seconds.

k. Exit the Add/Remove Programs Properties dialog box.

l. Exit the Control Panel.

m. Go to step 2.

2. Open the project `account.dsp` in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\oramts\samples\account.vc` directory using Microsoft Developer Studio.

3. Build `vcacct.dll`.

4. Back up the file `vcacct.dll` installed under *`ROOTDRIVE`*`:\program files\mts\samples\packages`.

5. Overwrite `vcacct.dll` with the one you built in step 3.

6. Choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Transaction Server > Bank Client.

   This starts the Sample Visual Basic bank application and runs `vbbank.exe` to test the component.

7. Click the Visual C++ radio button of the Language field.

8. Click the Account or the MoveMoney radio boxes under the Component field.

9. Enter the account number(s) and the amount for the operation.

10. Click Submit.

11. Start SQL*Plus:

```
C:\> sqlplus/NOLOG
```

12. Connect with the username `scott` and password `tiger` (unless you changed it):

    ```
    ENTER USER-NAME: scott/tiger
    ```

13. Verify the success of the operation:

    ```
    SQL> SELECT * FROM account;
    ```

    ```
    SQL> SELECT * FROM receipt;
    ```

Configuring Oracle ODBC Driver with the Microsoft Application Demo

You can use **Oracle Open Database Connectivity (ODBC) Driver** release 9.0 with the Microsoft sample application. See "Using the Oracle ODBC Driver" on page 5-26 for instructions on integrating the Oracle ODBC Driver with the sample application.

Configuring Oracle Provider for OLE DB with the Microsoft Application Demo

You can use **Oracle Provider for OLE DB** release 9.0 with the Microsoft sample application. See the *Oracle Provider for OLE DB Developer's Guide* for information about using Oracle Provider for OLE DB with Microsoft Transaction Server.

5

Programming with Microsoft Transaction Server and an Oracle Database Server

This chapter describes how to program with **Microsoft Transaction Server** and an Oracle database server.

This chapter contains these topics:

COM Component Integration in a Transaction Overview

The focal point of the transaction process is a component of Microsoft Transaction Server called **Microsoft Distributed Transaction Coordinator (MS DTC)**. When a client computer starts a business method on a transactional component, Microsoft Transaction Server begins a transaction coordinated by the MS DTC. The Oracle connection pooling layer enables the database server to act as a **resource manager (RM)** in the MS DTC-coordinated transaction. Figure 5–1 and Table 5–1 provide an overview of how these and other components perform a transaction.

Figure 5–1 Component Integration in a Transaction

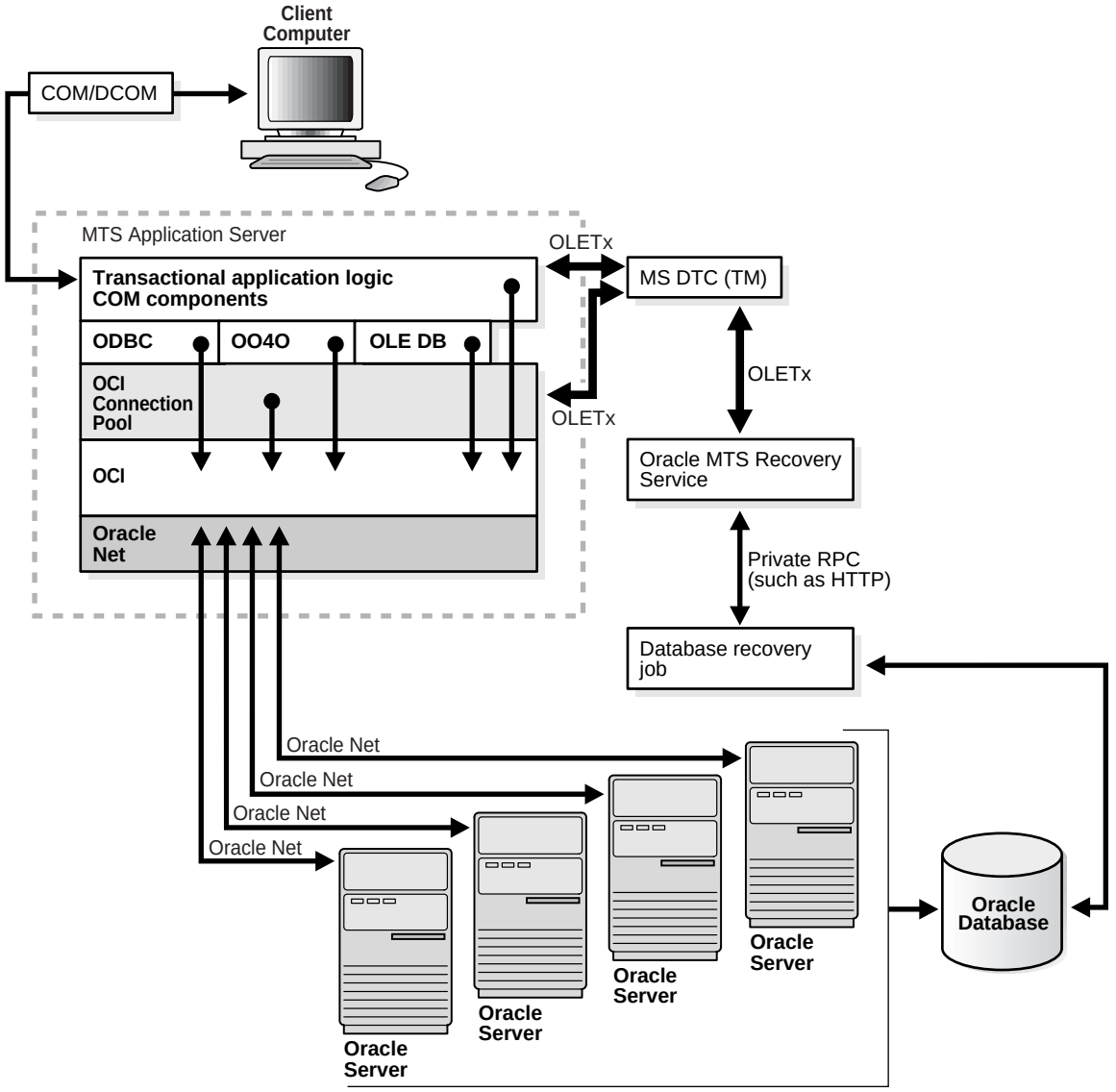


Table 5–1 Component Integration in a Transaction

Component	Major Responsibilities
Client computer connection	■ Activates the application components through a Web browser or through the component object model (COM) / distributed component object model (DCOM)
Transactional application logic COM components	■ Embed the business logic (If the component is transactional, Microsoft Transaction Server starts a transaction for every method invocation on that component.) ■ Acquire pooled connections to a database server through the Oracle resource dispenser and Oracle Call Interface (OCI), Oracle Open Database Connectivity (ODBC) Driver, Oracle Provider for OLE DB, or Oracle Objects for OLE (OO4O) ■ Decide the outcome of the operation by notifying Microsoft Transaction Server of its decision to commit or abort the changes to all RMs
Oracle ODBC Driver, OO4O, Oracle Provider for OLE DB, and OCI	■ Obtain a service context to the database server through the OCI connection pooling component ■ Provide connection pooling resources, if necessary (through Oracle Provider for OLE DB or Oracle ODBC Driver). The Oracle ODBC Driver provides pooled ODBC connections. Oracle Provider for OLE DB provides pooled data source objects. OO4O uses the OCI connection pool.
OCI connection pool	Performs the following for transaction components: ■ Enlists the RM (database server) in the component's Microsoft Transaction Server transaction ■ Starts an Oracle global transaction corresponding to the Microsoft Transaction Server transaction of which the component is a part ■ Acts as a resource dispenser to perform client-side connection pooling
Oracle MTS Recovery Service	■ Recovers Microsoft Transaction Server-related, in-doubt Oracle transactions that originated from the host computer
MS DTC (part of Microsoft Transaction Server)	■ Commits and aborts transactions using the two-phase commit protocol ■ Monitors transactions that require recovery. Multiple MS DTCs can be involved in a single transaction. When a transactional Microsoft Transaction Server component on computer A invokes another transactional Microsoft Transaction Server component on computer B, a connection is opened between the MS DTC on computer A and the MS DTC on computer B. When the root MS DTC commits or aborts a transaction, it sends the request through all involved MS DTCs. The transaction request is then passed to the OCI connection pooling / Microsoft Transaction Server integration, which sends it to the database server.
Database server	■ Acts as an RM for Microsoft Transaction Server. This is the database on which the client transaction request is performed.

Microsoft Transaction Server Application Development Overview

Regardless of the application program interface (API) you use, OCI connection pooling is used in nearly all cases to coordinate a transaction. Review the following sections for information about how a transaction is registered and OCI connection pooling coordinates the transaction:

- Microsoft Transaction Server Component Registration Overview
- Microsoft Transaction Server-Coordinated Component Transaction Overview
- MS DTC-Coordinated Component Transaction Overview

> **See Also:** *Microsoft Transaction Server Programmer's Guide* in the Microsoft Transaction Server's Help file for an explanation of how to develop application components for Microsoft Transaction Server

Microsoft Transaction Server Component Registration Overview

Application components that run in the Microsoft Transaction Server environment are created as dynamic link libraries (DLLs). Application components are registered with Microsoft Transaction Server using the Microsoft Transaction Server Explorer graphical user interface (GUI) tool. When you register the application component, you mark it as one of the types described in Table 5–2.

Table 5–2 Microsoft Transaction Server Component Registration

Type	The Component...
Requires a transaction	Must run in a transaction. If the transaction does not currently exist, Microsoft Transaction Server automatically creates a new transaction for each method invocation on the component.
Requires a new transaction	Must run within their own transaction. Microsoft Transaction Server automatically creates a new transaction for each method invocation on the component.
Supports transactions	Can run within the client's transaction. When a new component is created, its context inherits the transaction from the context of the invoking client. If the client does not have a transaction, the new context is also created without one.
Does not support transactions	Does not run within a transaction. Each method invocation on the component is performed without a surrounding transaction, regardless of whether the invoking client includes a transaction.

How you register an application component determines if it runs in a Microsoft Transaction Server-coordinated transaction. Table 5–3 describes this process.

Table 5–3 Running Components in a Microsoft Transaction Server Transaction

If the Application Component...	Then...
Runs in a Microsoft Transaction Server-coordinated transaction	OCI connection pooling is always used and Microsoft Transaction Server and its MS DTC component coordinate the creation, startup, management, and commitment phases of the transaction. Microsoft Transaction Server ensures that all changes made by the component are committed if the transaction succeeds, or are aborted if the transaction fails. **See Also:** "Microsoft Transaction Server-Coordinated Component Transaction Overview" on page 5-7
Does not run in a Microsoft Transaction Server-coordinated transaction	The component runs in a Microsoft Transaction Server environment, but the database servers that it accesses may or may not take part in MS DTC-coordinated transactions. If the transaction is not MS DTC-coordinated, the client application must create, start, manage, and commit the transaction. OCI connection pooling may be used, depending upon the interface accessing the database server (such as Oracle Provider for OLE DB, Oracle ODBC Driver, OO4O, or others). **See Also:** "MS DTC-Coordinated Component Transaction Overview" on page 5-8

Microsoft Transaction Server-Coordinated Component Transaction Overview

This section describes how OCI connection pooling, Microsoft Transaction Server, and MS DTC operate with application components in a Microsoft Transaction Server-coordinated transaction environment.

1. The client API being used (Oracle ODBC Driver, OCI, OO4O, or Oracle Provider for OLE DB) calls OCI function `OraMTSSvcGet()` to obtain a service context from the OCI connection pooling component.
2. The OCI connection pooling component enlists the transaction, which is to be coordinated by the MS DTC component of Microsoft Transaction Server.

   These actions return OCI service and environment handles to client applications.
3. The client application:
   a. Performs the database operations.
   b. Calls OCI function `OraMTSSvcRel()` to release the OCI pooling connection obtained at the beginning of the transaction.
   c. Calls `SetComplete` (to commit database operations) or `SetAbort` (to abort database operations) on the Microsoft Transaction Server context object associated with the component.
4. MS DTC performs the two-phase commit protocol to prepare and commit (or abort) the transaction, which notifies the OCI connection pooling component and ends the transaction.
5. OCI connection pooling is notified and performs the necessary steps to complete phase one (the prepare phase) and phase two (the commit or abort phase).

MS DTC-Coordinated Component Transaction Overview

This section describes how OCI connection pooling, Microsoft Transaction Server, and MS DTC operate with application components not running in a Microsoft Transaction Server-coordinated transaction, but using MS DTC.

1. The client application starts an MS DTC transaction and connects to the database server. This is either a:
   - Nonpooled OCI connection obtained through OCI logon calls (for example, `OCIServerAttach()` and `OCISessionBegin()`). For these connections, the application calls `OraMTSEnlCtxGet()` to associate the OCI service context with an Microsoft Transaction Server enlistment context.
   - Connection pool obtained by calling `OraMTSSvcGet(..,..,ORAMTS_CFLG_NOIMPLICIT)`, and not yet released with `OraMTSSvcRel()`
2. For nonpooled connections, the client application passes in the enlistment context to `OraMTSJoinTxn()`.
3. For pooled connections, the client application passes the OCI service context into `OraMTSSvcEnlist()`.
4. The OCI connection pooling component enlists the connection (pooled or nonpooled) in the transaction coordinated by the MS DTC component of Microsoft Transaction Server.
5. The client application:
   a. Performs database operations.
   b. Calls `OraMTSSvcEnlist()` with a `NULL` transaction reference to de-enlist from an MS DTC coordinated transaction. For nonpooled connections, `OraMTSTxnJoin()` is invoked with a `NULL` transaction reference to perform the de-enlistment.
   c. Calls `OraMTSSvcRel()` to release a pooled connection back to the pool. For nonpooled connections, the client calls `OraMTSEnlCtxRel()` to release the enlistment context and then logs off the database.
   d. Calls the commit or abort method on the MS DTC transaction object (for example, `pTransaction->Commit()` or `pTransaction->Abort()`).
6. MS DTC performs the two-phase commit protocol to commit the transaction.
7. OCI connection pooling is notified and performs the necessary steps to complete phase one (the prepare phase) and phase two (the commit or abort phase).

OCI Integration with Microsoft Transaction Server Overview

The following OCI functions enable you to integrate the OCI client application with Microsoft Transaction Server and a database server. Review the following sections for information about this integration:

OCI and Microsoft Transaction Server Function Overview

You must use OCI release 8.1 or higher. OCI releases earlier than 8.1 are not supported.

Caution: As with any C++ Microsoft Transaction Server component, obtain the object context and call `SetAbort()`, `SetComplete()`, `EnableCommit()`, or `DisableCommit()`, depending on the state of the component's work. Do *not* make any OCI transaction calls such as `OCITransCommit()` or `OCITransAbort()`; this corrupts the data!

The only code change to make is in obtaining and releasing the OCI service context handle. An OCI service context handle and environment handle are acquired when you obtain a pooled OCI connection to the database with the OCI function `OraMTSSvcGet()`. Include the `oramts.h` header and link with the `oramts.lib` library. When you are finished, call OCI function `OraMTSSvcRel()` to release the service context handle and environment handle. Using `OraMTSSvcGet()` enables you to receive connection pooling and implicit transaction support (if you registered the application component to run in a Microsoft Transaction Server

transaction). See "OraMTSSvcGet() Function" on page 5-13 and "OraMTSSvcRel() Function" on page 5-16 for more information.

In releases prior to 9.0, `OraMTSSvcEnlist()` and `OraMTSSvcEnlistEx()` enlisted nonpooled OCI connections in Microsoft Transaction Server transactions. This is no longer supported. These functions are still available to enlist pooled connections in MS DTC-coordinated transactions. To enlist nonpooled OCI connections in Microsoft Transaction Server-started and MS DTC-coordinated transactions, the client must use `OraMTSJoinTxn()`. See "OraMTSJoinTxn() Function" on page 5-23 for more information.

Ensure that for each process, you call `OCIInitialize` at least once before executing any other OCI calls. This initializes the OCI process environment. In addition, you must pass it the `OCI_THREADED` flag. If you are using Microsoft's Internet Information Server (IIS) and the components are being called as in-process libraries, then `OCIInitialize` is already called for you.

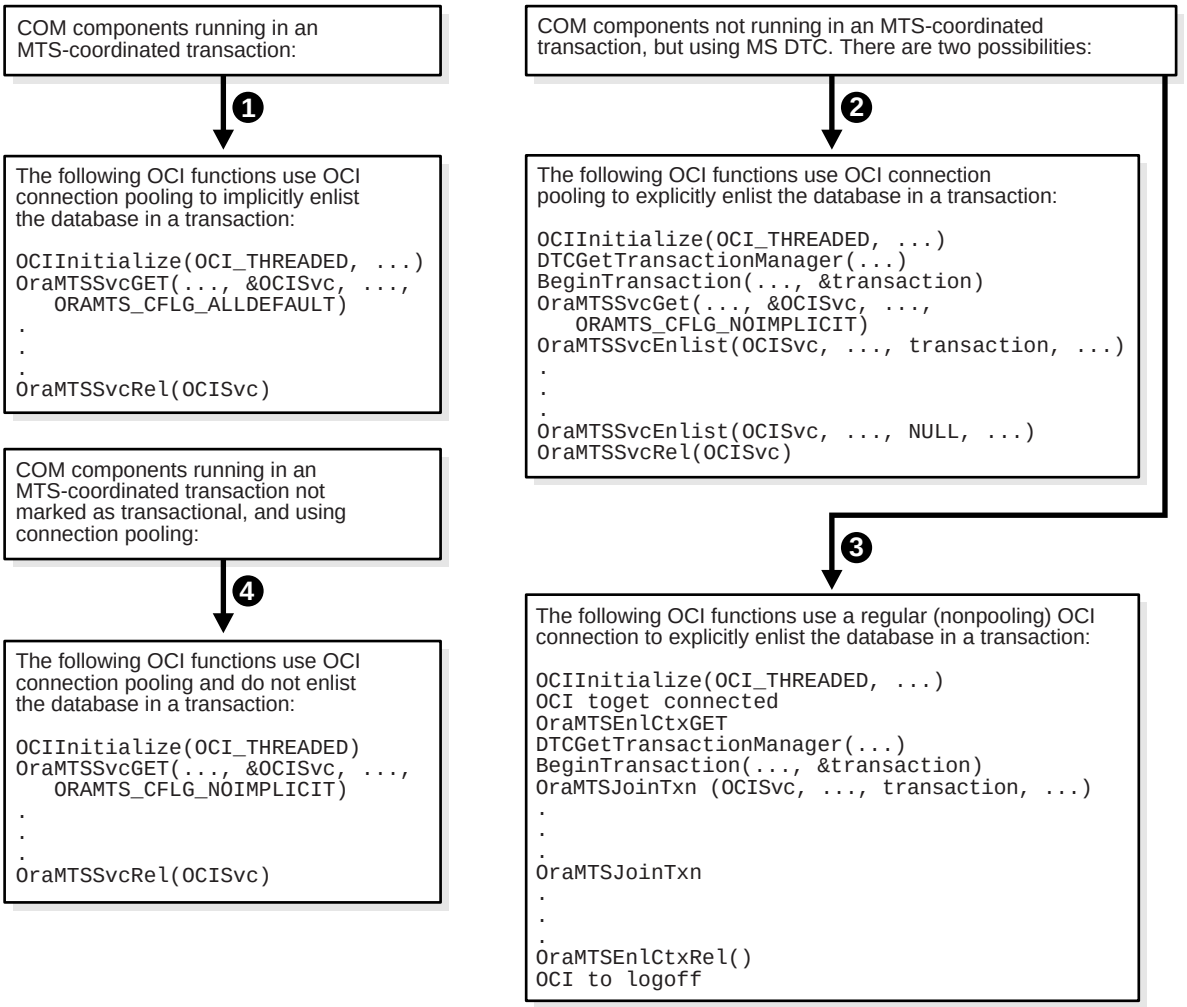
```
#include <oci.h>
#include  <oramts.h>
#include  <xolehlp.h>
// other MTS relevant includes ...

// prototype for the error handler.
BOOL Chekerr(sword swOCIStat, OCIError *OCIErrh);

// MTS component method
HRESULT OCITestMethod()
{
 IObjectContext *pObjectContext = NULL;
 OCIEnv    *myenvh = NULL;
 OCISvcCtx *mysvch = NULL;
 OCIError  *myerrh = NULL;
 OCIStnt   *mystmh = NULL;
 DWORD      dwStat;
 HRESULT    hRes = S_OK;
 sword      swOCIStat;
 BOOL       bCommit = FALSE;
 char      *lpzStmt = "UPDATE EMP SET SAL = SAL + 1000";

 // Initialize the OCI environment first -- request OCI_THREADED
 OCIInitialize(OCI_THREADED, (dvoid*)NULL,NULL,NULL,NULL);
 // attempt to get a connection to the database through the resource dispenser
```

```
 OraMTSSvcGet(
"scott","tiger","finprod_db",&mysvch, &myenvh, ORAMTS_CFLG_ALLDEFAULT);
 // validate return status
 if(dwStat != ORAMTS_ERR_NOERROR)
 {
   printf("error: failed to obtain a connection to the database - %ld",
dwStat);
   goto cleanup;
 }
 // successful logon and enlistment in the MTS transaction. allocate statement
 // handles and other handles using the OCI environment handle myenvh ....
 swOCIStat = OCIHandleAlloc(myenvh, (void *)&myerrh,OCI_HTYPE_ERROR, 0 , NULL);
 if (Checkerr(swOCIStat, myerrh)) goto cleanup;
 swOCIStat = OCIHandleAlloc(myenvh, (dvoid *)&mystmh,OCI_HTYPE_STMT, 0,NULL);
 if (Checkerr(swOCIStat, myerrh)) goto cleanup;
 // prepare a DML statement
 OCIStmtPrepare(mystmh, myerrh, lpzStmt, lstrlen(lpzStmt), OCI_NTV_SYNTAX,
OCI_DEFAULT)
 Checkerr(swOCIStat, myerrh);
 // execute the statement -- ensure that AUTOCOMMIT is not requested.
 OCIStmtExecute(mysvch, mystmh, myerrh, 1, 0, NULL, NULL, OCI_DEFAULT);
 if (Checkerr(swOCIStat, myerrh)) goto cleanup;
 // all's well so far choose to go for a commit
 bCommit = TRUE;
cleanup:
 if (mystmh) OCIHandleFree((void*)mystmh, OCI_HTYPE_STMT);
 if (myerrh  OCIHandleFree((void*)myerrh, OCI_HTYPE_ERROR);
 if (mysvch) OraMTSSvcRel(mysvch);
 if (bCommit)
     pObjectContext->SetComplete();
 else
     pObjectContext->Abort();
 return(bCommit ? S_OK : E_FAIL);
}
```

See Also: The files in the *ORACLE_BASE**ORACLE_HOME*\oramts\samples\account.vc directory for code samples

Figure 5–2 provides a high-level overview of how to use the OCI functions `OraMTSSvcGet()`, `OraMTSSvcRel()`, and `OraMTSJoinTxn`.

Figure 5–2 Using OCI Functions



COM applications not hosted by the Microsoft Transaction Server environment (also known as standalone applications) can also use methods 2, 3, and 4 in Figure 5–2. However, these application types cannot use declarative transactions (through the Microsoft Transaction Server Explorer Microsoft Management Console snap-in).

OraMTSSvcGet() Function

Purpose

`OraMTSSvcGet()` obtains a pooled connection (also known as an OCI service context) from the OCI connection pool. The pooled connection includes an OCI service context handle and OCI environment handle.

Syntax

```
DWORD  OraMTSSvcGet(
                      text*       lpUname,
                      text*       lpPsswd,
                      text*       lpDbnam,
                      OCISvcCtx** pOCISvc,
                      OCIEnv**    pOCIEnv,
                      ub4         dwConFlgs
                    );
```

Parameters

Table 5–4 describes the `OraMTSSvcGet()` parameters.

Table 5–4 OraMTSSvcGet() Parameters

Datatype	Parameter	Description
`text*`	`lpUname(IN)`	Username for connecting to the database server
`text*`	`lpPsswd(IN)`	Password for the username
`text*`	`lpDbnam(IN)`	The **net service name** for connecting to the database (created with Oracle Net Manager or Oracle Net Configuration Assistant)
`OCISvcCtx**`	`pOCISvc(OUT)`	Pointer to the OCI service context handle
`OCIEnv**`	`pOCIEnv(OUT)`	Pointer to the OCI environment handle
`ub4`	`dwConFlgs(IN)`	Connection flags, for which there are the following possible values: ▪ `ORAMTS_CFLG_ALLDEFAULT` Obtains a pooled connection and enlists the connection in any Microsoft Transaction Server transaction, if one exists. If the component is nontransactional, no enlistment request is dispensed.

Table 5–4 OraMTSSvcGet() Parameters

Datatype	Parameter	Description
		■ ORAMTS_CFLG_NOIMPLICIT Obtains a pooled connection, but does not enlist the resource in any Microsoft Transaction Server transaction even if the component is transactional. Use this flag if the component enlists the connection resource later using OraMTSSvcEnlist(). Prior to releasing a connection obtained in this fashion, the client must de-enlist the resource if enlisted. ■ ORAMTS_CFLG_UNIQUESRVR Requests a single OCI session for each OCI Server. In this release, multiplexing is not supported. Therefore, this option is always used. ■ ORAMTS_CFLG_SYSDBALOGN Use this flag if connecting as SYSDBA. ■ ORAMTS_CFLG_SYSOPRLOGN Use this flag if connecting as SYSOPER. ■ ORAMTS_CFLG_PRELIMAUTH Use this flag if connecting as the user INTERNAL to pre-Oracle9*i* databases. The INTERNAL account is no longer valid as of Oracle9*i*. Instead, log on with a SYSDBA or SYSOPER account using the ORAMTS_CFLG_SYSOPRLOGN or ORAMTS_CFLG_SYSDBALOGN flag.

Returns

Returns ORAMTSERR_NOERROR upon successful acquisition of an OCI pooling connection (OCI service context).

Comments

OraMTSSvcGet() returns a pooled OCI connection to the caller, enabling a database transaction using OCI to begin. Use OraMTSSvcGet() to implicitly enlist the OCI connection in a transaction coordinated by Microsoft Transaction Server. In this type of transaction, Microsoft Transaction Server controls the creation, startup, management, and commitment phases of the transaction through its MS DTC component.

`OraMTSSvcGet()` also provides connection pooling without enlisting the database server in a Microsoft Transaction Server transaction. This is done by setting `OraMTSSvcGet()` as follows:

```
OraMTSSvcGet(...,ORAMTS_CFLG_NOIMPLICIT)
```

In all cases where `OraMTSSvcGet()` is used, you must always use `OraMTSSvcRel()` to release the connection when finished.

Note: Connection pooling is used regardless of whether you enlist or do not enlist the COM component in a transaction.

Use the flags `ORAMTS_CFLG_SYSDBALOGN` and `ORAMTS_CFLG_SYSOPRLOGN` when connecting as `SYSDBA` and `SYSOPER`, respectively.

To obtain a nonenlisted connection using the `scott/tiger` account, call `OraMTSSvcGet()` as follows:

```
OraMTSSvcGet("SCOTT", "TIGER", "oracle", &OCISvc, &OCIEnv, ORAMTS_CFLG_
ALLDEFAULT | ORAMTS_CFLG_NOIMPLICIT);
```

`OraMTSSvcGet()` does not support placing the username (`lpUname`), password (`lpPsswd`), and net service name syntax (`lpDbname`) together in the username argument (for example, `scott/tiger@prod_fin`). Instead, the caller must fill in `lpUname`, `lpPsswd`, and `lpDbname` separately (as shown in the previous syntax example). Calling `OraMTSSvcGet()` with the username and password as `NULL` strings uses external authentication (operating system authentication) for the connection.

OraMTSSvcRel() Function

Purpose

`OraMTSSvcRel()` releases a pooled OCI connection (OCI service context) back to the connection pool. Use `OraMTSSvcRel()` to release connections that were acquired with `OraMTSSvcGet()`.

Syntax

```
DWORD OraMTSSvcRel(OCISvcCtx* OCISvc);
```

Parameters

Table 5–5 describes the `OraMTSSvcRel()` parameters.

Table 5–5 OraMTSSvcRel() Parameters

Datatype	Parameter	Description
`OCISvcCtx*`	`OCISvc(IN)`	OCI service context for a pooled connection

Returns

Returns `ORAMTSERR_NOERROR` upon successful release of a pooled OCI connection.

Comments

An OCI pooled connection obtained through a previous call to `OraMTSSvcGet()` is released back to the connection pool. Once released back to the connection pool, the OCI service context, its environment handle, and all child handles are invalid.

A nontransactional client component must explicitly call `OCITransCommit()` or `OCITransAbort()` prior to releasing a connection obtained through `OraMTSSvcGet(…,…,ORAMTS_CFLG_ALLDEFAULT)` back to the pool. Otherwise, all changes made in that session are rolled back. A transaction component uses the `SetComplete` or `SetAbort` methods on its Microsoft Transaction Server object context.

Components that have called `OraMTSSvcGet(…,…,ORAMTS_CFLG_NOIMPLICIT)` to obtain a connection resource must first de-enlist the resource if enlisted. If the connection was enlisted explicitly, `pTransaction->Commit()` or `pTransaction->Abort()` must be called. Otherwise, `OCITransCommit()` or `OCITransAbort()` must be called before releasing the connection back to the pool.

OraMTSSvcEnlist() Function

Purpose

`OraMTSSvcEnlist()` enlists or de-enlists an OCI connection in a transaction coordinated by MS DTC. Use this call to explicitly enlist pooled connections. Nonpooled connections must enlist with `OraMTSJoinTxn()`.

Syntax

```
DWORD OraMTSSvcEnlist(
                       OCISvcCtx*  OCISvc,
                       OCIError*   OCIErr,
                       void*       lpTrans,
                       unsigned    dwFlags
                      );
```

Parameters

Table 5–6 describes the `OraMTSSvcEnlist()` parameters.

Table 5–6 OraMTSSvcEnlist() Parameters

Datatype	Parameter	Description
`OCISvcCtx*`	`OCISvc(IN)`	OCI service context for pooled connections obtained by calling `OraMTSSvcGet()`
`OCIError*`	`OCIErr(IN/OUT)`	OCI error handle (ignored)
`void*`	`lpTrans(IN)`	Pointer to the MS DTC-controlled transaction in which to enlist. If `NULL`, the OCI connection is de-enlisted from the MS DTC-controlled transaction.
`unsigned`	`dwFlags(IN)`	Flag used for enlisting in a transaction. Use the `ORAMTS_ENFLG_DEFAULT` value. If enlisting, then start a new Oracle global transaction. If de-enlisting, then detach from any global Oracle transaction and delete the context object if the OCI service context represents a nonpooled connection.

Returns

Returns `ORAMTSERR_NOERROR` on success.

Comments

Use this call to explicitly enlist or de-enlist a pooled connection. For enlisting and de-enlisting nonpooled connections, use `OraMTSSvcRel()`.

`OraMTSSvcEnlist()` enlists (or de-enlists) pooled OCI connections obtained previously through `OraMTSSvcGet()` with the `ORAMTS_CFLG_NOIMPLICIT` flag and not yet released with `OraMTSSvcRel()`. The pooled OCI connections must be explicitly enlistable. When the transaction is complete, you must de-enlist `OraMTSSvcEnlist()`, passing `NULL` as the transaction pointer as follows:

```
OraMTSSvcEnlist(OCISvc, OCIErr, NULL, ORAMTS_ENFLG_DEFAULT)
```

You must use `OraMTSSvcRel()` to release the connection when done.

Callers must:

1. Allocate a connection.
2. Enlist the connection.
3. Perform work.
4. De-enlist the connection.
5. Release the connection.
6. Attempt to commit or abort.

OraMTSSvcEnlistEx() Function

Purpose

`OraMTSSvcEnlistEx()` enlists an OCI connection or service context in an MS DTC transaction. Use this call only to explicitly enlist pooled connections. Nonpooled connections must enlist with `OraMTSJoinTxn()`.

Syntax

```
DWORD OraMTSSvcEnlistEx(
                          OCISvcCtx* OCISvc,
                          OCIError*  OCIErr,
                          void*      lpTrans,
                          unsigned   dwFlags,
                          char*      lpDBName
                        );
```

Parameters

Table 5–7 describes the `OraMTSSvcEnlistEx()` parameters.

Table 5–7 OraMTSSvcEnlistEx() Parameters

Datatype	Parameter	Description
`OCISvcCtx*`	`OCISvc`	OCI service context for a pooled connection obtained by calling `OraMTSSvcGet()`
`OCIError*`	`OCIErr`	OCI error handle (ignored)
`void*`	`lpTrans`	Pointer to an MS DTC-controlled transaction. If `NULL`, the OCI connection is de-enlisted from the MS DTC-controlled transaction.
`unsigned`	`dwFlags`	Enlistment flag, for which there is the `ORAMTS_ENFLG_DEFAULT` value. If enlisting, then start a new Oracle global transaction. If de-enlisting, then exit any global Oracle transaction and delete the context object if the OCI service context represents a nonpooled connection.
`char*`	`lpDBName`	Net service name for connecting to the database (created with Oracle Net Manager or Oracle Net Configuration Assistant)

Returns

Returns ORAMTSERR_ILLEGAL_OPER.

Comments

Use OraMTSSvcEnlistEx() for pooled connections or OraMTSJoinTxn() for nonpooled connections.

OraMTSEnlCtxGet() Function

Purpose

OraMTSEnlCtxGet() creates an enlistment context for a nonpooled OCI connection.

Syntax

```
DWORD OraMTSEnlCtxGet(
                        text*       lpUname,
                        text*       lpPsswd,
                        text*       lpDbnam,
                        OCISvcCtx*  pOCISvc,
                        OCIError*   pOCIErr,
                        ub4         dwFlags,
                        void**      pCtxt
                      );
```

Parameters

Table 5–8 describes the OraMTSEnlCtxGet() parameters.

Table 5–8 OraMTSEnlCtxGet() Parameters

Datatype	Parameter	Description
text*	lpUname (IN)	Username for connecting to the database server
text*	lpPsswd (IN)	Password for connecting to the database server
text*	lpDbnam (IN)	Net service name for connecting to a database server
OCISvcCtx*	pOCISvc (IN)	OCI service context for a nonpooled connection
OCIError*	pOCIErr (IN)	OCI error handle
ub4	dwFlags(IN)	Enlistment flags. The only value currently permitted is 0.
void**	pCtxt (OUT)	Enlistment context to be created

Returns

Returns ORAMTSERR_NOERROR on success.

Comments

This call sets up an enlistment context for a nonpooled connection. This call must be started just after the caller establishes the OCI connection to the database. Once created, this context can be passed into OraMTSJoinTxn() calls. Prior to deleting the OCI connection, OraMTSEnlCtxRel() must be called to delete the enlistment context.

Callers must:

1. Allocate a nonpooled connection through OCI.
2. Create an enlistment context by calling OraMTSEnlCtxGet().
3. Enlist the connection by calling OraMTSJoinTxn().
4. Perform database work.
5. De-enlist the connection by calling OraMTSJoinTxn() with a NULL transaction pointer.
6. Attempt to commit or abort work.
7. Release the enlistment context by calling OraMTSEnlCtxRel().
8. Release the nonpooled OCI connection and delete its associated OCI environment handle.

OraMTSEnlCtxRel() Function

Purpose

`OraMTSEnlCtxRel()` eliminates a previously set up enlistment context for a nonpooled OCI connection.

Syntax

```
DWORD OraMTSEnlCtxRel(void* pCtxt);
```

Parameters

Table 5–9 describes the `OraMTSEnlCtxRel()` parameters.

Table 5–9 OraMTSEnlCtxRel() Parameters

Datatype	Parameter	Description
`void*`	`pCtxt (IN)`	Enlistment context to eliminate

Returns

Returns `ORAMTSERR_NOERROR` on success.

Comments

Before dropping a nonpooled OCI connection, a client must call `OraMTSEnlCtxRel()` to eliminate any enlistment context it may have created for that connection. The enlistment context can maintain OCI handles allocated off the connection's OCI environment handle. This makes it imperative that the environment handle is not deleted for the associated enlistment context.

OraMTSJoinTxn() Function

Purpose

`OraMTSJoinTxn()` enlists a nonpooled OCI connection in an MS DTC transaction.

Syntax

```
DWORD OraMTSJoinTxn(void*  pCtxt, void*  pTrans);
```

Parameters

Table 5–10 describes the `OraMTSJoinTxn()` parameters.

Table 5–10 OraMTSJoinTxn() Parameters

Datatype	Parameter	Description
`void*`	`pCtx (IN)`	Enlistment context for the OCI connection
`void*`	`pTrans (IN)`	Reference to the MS DTC transaction object

Returns

Returns `ORAMTSERR_NOERROR` on success.

Comments

Clients use this call with nonpooled OCI connections to enlist connections in MS DTC-coordinated transactions. The client passes in the wide reference to the enlistment context representing the OCI connection, along with a reference to an MS DTC transaction object. If `pTrans` is `NULL`, the OCI connection is de-enlisted from any MS DTC transaction in which it is currently enlisted. You can enlist a previously-enlisted OCI connection in a different MS DTC transaction.

OraMTSTransTest() Function

Purpose

`OraMTSTransTest()` tests if you are running inside a Microsoft Transaction Server-started transaction.

Syntax

```
BOOL OraMTSTransTest();
```

Parameters

None.

Returns

Returns `true` if running inside a Microsoft Transaction Server transaction. Otherwise, `false` is returned.

Comments

Microsoft Transaction Server transactional components use `OraMTSTransTest()` to check if a component is running within the context of a Microsoft Transaction Server transaction. Note that this call can only test Microsoft Transaction Server-started transactions. Transactions started by directly calling the MS DTC are not detected.

OraMTSOCIErrGet() Function

Purpose

OraMTSOCIErrGet() retrieves the OCI error code and message text (if any) from the last OraMTS function operation, typically OraMTSSvcGet() or OraMTSJoinTxn().

Syntax

```
BOOL OraMTSOCIErrGet(DWORD* dwErr, LPCHAR lpcEMsg, DWORD* lpdLen);
```

Parameters

Table 5–11 describes the OraMTSOCIErrGet() parameters.

Table 5–11 OraMTSOCIErrGet() Parameters

Datatype	Parameter	Description
DWORD*	dwErr	Error code
LPCHAR	lpcEMsg	Buffer for the error message, if any
DWORD*	lpdLen	Set to the actual number of message bytes

Returns

Returns true if an OCI error is encountered. Otherwise, false is returned. If true is returned and lpcEMsg and lpdLen are valid, and there is a stashed error message, up to lpdLen bytes are copied into lpcEMsg. lpdLen is set to the actual number of message bytes.

Comments

OraMTSOCIErrGet() retrieves the OCI error code and OCI error message text, if any, from the last OraMTSSvc operation on this thread. For example:

```
DWORD dwStat = OraMTSSvcGet("scott", "invalid_password","fin_prod",
db",&mysvch, &myenvh, ORAMTS_CFLG_ALLDEFAULT);
        if (dwStat != ORAMTS_ERR_NOERROR)
        {
                DWORD   dwOCIErr;
                char    errBuf[MAX_PATH];
                DWORD   errBufLen = sizeof(effBuf);

                if (OraMTSOCIErrGet(&dwOCIErr, &errBuf, &errBufLen))
                        printf("OCIError %d: %s"\n);
        }
```

ODBC Integration with Microsoft Transaction Server Overview

This section describes how to use Oracle ODBC Driver with Microsoft Transaction Server and a database server. Specific topics discussed are:

- Setting the Connection Attribute
- Using the Oracle ODBC Driver (recommended)
- Using Microsoft's Oracle ODBC Driver

OCI connection pooling operates as described in "Microsoft Transaction Server Application Development Overview" on page 5-5, with no changes to OCI code required for ODBC to operate.

Setting the Connection Attribute

To use Microsoft Transaction Server with either Oracle ODBC Driver 9.0 or Microsoft's Oracle ODBC driver, you must set the connection attribute. Use the function `SQLSetConnectAttr` to call the parameter `SQL_ATTR_ENLIST_IN_DTC` in the ODBC code. This enables you to receive connection pooling and implicit transaction support. See "Setting Up MTS to Access Oracle" in the Microsoft Transaction Server online Help for instructions.

Using the Oracle ODBC Driver

The ODBC Driver Manager distributed with ODBC 3.0 is a Resource Dispenser that supports connection pooling. (See the Microsoft Transaction Server SDK for information.) Oracle ODBC Driver release 9.0 integrates with the ODBC 3.0 Driver Manager by supporting the `SQLSetConnectAttr(...,..., SQL_ATTR_ENLIST_IN_DTC)` call to enlist or de-enlist the ODBC connection in or from MS DTC-coordinated transactions.

Use the Oracle ODBC Driver 9.0 with:

- Applications you develop
- The sample banking application that Microsoft provides with Microsoft Transaction Server (See Chapter 4, "Running the Microsoft Application Demo" for more information.)

To configure Oracle ODBC Driver 9.0:

1. Choose Start > Settings > Control Panel.

   The Control Panel window appears.

2. Double-click ODBC.

   The ODBC Data Source Administrator dialog box appears.

3. Choose the File DSN tab.

4. To make Oracle's ODBC Driver work with Microsoft's sample banking application demo, follow Substeps 4a through 4d. Otherwise, go to Step 5.

   a. Back up Microsoft's `mtssamples.dsn` file. This file is located in *`ROOTDRIVE`*`:\program files\common files\odbc\data sources`.

   b. Select `mtssamples.dsn` and click Remove.

   c. Click Yes when prompted.

      This deletes the configuration file that enables the Microsoft Transaction Server sample application demo to use Microsoft's ODBC driver.

   d. Go to step 5.

5. Click Add to create a new File data source name (DSN).

   The Create New Data Source wizard appears.

6. Select Oracle ODBC Driver 9.0.

7. Click Advanced.

8. Add the following information in the keywords and values field:

   ```
   SERVER=database_alias
   USERNAME=scott
   PASSWORD=tiger
   ```

   The following table describes the keywords and values.

Where...	Is...
SERVER	The database alias used by the demo to access the database server (`mtsdemo`)
USERNAME	`scott` (database server username for this application)
PASSWORD	`tiger` (database server password for username `scott`, unless you changed it)

9. Click OK.
10. Click Next to continue with the Create New Data Source wizard.
11. Review the following table and enter the name of the file DSN to which to save this connection information:

If Using Oracle's ODBC For...	Then Enter...
Microsoft's sample application	`mtssamples.dsn` (Microsoft's ODBC name). This name must exactly match the name you removed in Substep 4b.
Applications you develop	Any appropriate name

12. Complete the remaining Create New Data Source wizard pages.
13. Click OK to exit the ODBC Data Source Administrator dialog box.
14. Exit the Control Panel window.

Using Microsoft's Oracle ODBC Driver

If the database server release is 8.0.5 or earlier, you cannot use the integration information described in this guide. However, there is a solution if you use Microsoft's Oracle ODBC driver. No other APIs are supported.

You can use Microsoft's Oracle ODBC Driver included in Windows NT Option Pack 4 to enable applications to interact with Microsoft Transaction Server and a database server. If you use this driver, the rest of the information in this guide does not apply and you do not receive the following:

- Performance benefits
- Other API support of Oracle integration
- Oracle 9.0 client support

See "Setting Up MTS to Access Oracle" in the Microsoft Transaction Server online Help for instructions on enabling Microsoft's Oracle ODBC Driver. After following those instructions, perform these additional steps:

To configure Microsoft's Oracle ODBC Driver:

1. Install Oracle Required Support Files (RSF) release 7.3.4 and SQL*Net 2.3 on the computer where Microsoft's Oracle ODBC Driver is operating.
2. Run the *ORACLE_BASE**ORACLE_HOME*\oramts\samples\sql\omtssamp.sql script.
3. Use SQL*Net Easy Config to set up a database alias connection. This is the alias that the mtssamples.dsn file uses.
4. If you installed the release 7.3.4 RSFs in a home with Oracle Net installed, be sure to set the following registry parameter at HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE:

   ```
   ORAOCI = ORA73.DLL
   ```

OO4O Integration with Microsoft Transaction Server Overview

There are no special requirements for using OO4O. You must use release 8.1 or higher of OO4O. See the OO4O online Help file for additional information about using OO4O with Microsoft Transaction Server.

> **See Also:** "Microsoft Transaction Server Application Development Overview" on page 5-5 for information about how connection pooling operates (with no changes required to the OO4O code)

Oracle Provider for OLE DB Integration with Microsoft Transaction Server Overview

See the *Oracle Provider for OLE DB Developer's Guide* for information about using Oracle Provider for OLE DB with Microsoft Transaction Server.

Other API Integration with Microsoft Transaction Server Overview

Other APIs are not currently supported, unless they use Oracle's ODBC Driver 8.1 or higher, such as ActiveX Data Objects.

6

Tuning Microsoft Transaction Server Performance

This chapter provides **Microsoft Transaction Server** performance tuning information:

This chapter contains these topics:

Improving Microsoft Transaction Server Application Performance

You can improve performance when you optimize the programming methods. Placing all code for a given transaction into one **component object model (COM)** component means you do not mark that component as transactional. This eliminates the overhead of going through Microsoft Transaction Server.

You can then use the Oracle commit or rollback functions to control that transaction in the component. If you are using the **Oracle Call Interface (OCI)**, you can still use `ORAMTSSvcGet()`, but you can also use the `ORAMTS_CFLG_NOIMPLICIT` flag. If you are updating across two or more Oracle database servers, use database links and connect to one database from the COM component.

> **See Also:** "OCI Integration with Microsoft Transaction Server Overview" on page 5-9 for more information about using `ORAMTSSvcGet()`

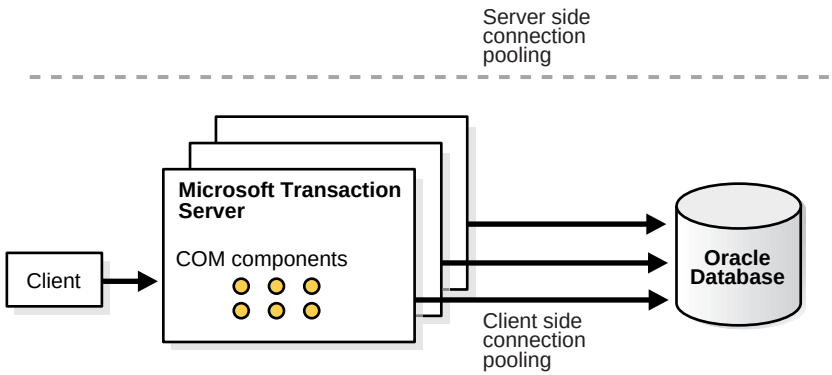
Managing Microsoft Transaction Server Connections

When a COM component ends a session with the database server, the connection by default does not immediately terminate. Instead, the connection remains idle in a connection pool, where it is available for reuse by another COM component attempting a new connection to the database server.

The idle period during which a connection is reusable reduces the resource costs associated with opening a new connection. The amount of time that the connection remains idle and available in the connection pool is determined by several registry parameter settings. You can modify these parameters on the computers on which the client Microsoft Transaction Server components are installed.

Figure 6–1 identifies the connection pool locations and the registry parameters associated with the client side pool.

Figure 6–1 Connection Pool Registry Parameters



Client side connection pooling registry parameters:

ORAMTS_CONN_POOL_TIMEOUT
ORAMTS_NET_CACHE_TIMEOUT
ORAMTS_NET_CACHE_MAXFREE
ORAMTS_OSCREDS_MATCH_LEVEL

Client side parameters are set in HKEY_LOCAL_
MACHINE\SOFTWARE\ORACLE\HOME*ID*.

Table 6–1 describes the connection pool registry parameters.

Table 6–1 Connection Pool Registry Parameters

Client Side Parameter	Description	Default Value Entry
ORAMTS_CONN_POOL_TIMEOUT	This parameter enables you to set how long a connection remains idle and available for reuse in the client side connection pool before timing out. After timing out, the connection is released.	120 seconds
ORAMTS_NET_CACHE_TIMEOUT	The portion of the connection associated with ORAMTS_CONN_POOL_TIMEOUT is responsible for session issues such as username and password (that is, the logon session). The portion of the connection associated with ORAMTS_NET_CACHE_TIMEOUT is responsible for communication issues such as sending and receiving data (that is, the actual Oracle Net connection). Establishing a new Oracle Net connection requires more resources than using that connection to establish a logon session. It is advisable to keep this value set higher then the session timeout value associated with ORAMTS_CONN_POOL_TIMEOUT. After ORAMTS_CONN_POOL_TIMEOUT times out, the portion of the connection associated with ORAMTS_NET_CACHE_TIMEOUT remains available for a slightly longer period of time. A server connection can then be reused by creating a new session with it.	120 seconds **Note:** This value is in addition to the value you set for ORAMTS_CONN_POOL_TIMEOUT. For example, if you set ORAMTS_CONN_POOL_TIMEOUT to 180, and set ORAMTS_NET_CACHE_TIMEOUT to 60, the time period before a connection completely terminates is 240 seconds.
ORAMTS_NET_CACHE_MAXFREE	This parameter enables you to set the maximum number of free server connections to maintain in the client side connection pool at any given time.	5

Table 6–1 Connection Pool Registry Parameters

Client Side Parameter	Description	Default Value Entry
ORAMTS_OSCREDS_MATCH_LEVEL	This parameter enables you to set the degree of Windows NT security checking to perform on a connection when the OS_ROLES initialization parameter is set to true in the init.ora file. When a user connects to the database server (for example, with the CONNECT / command), there are certain database roles and privileges associated with their Windows NT username. When the user disconnects, the connection becomes idle and available in the pool. When another user enters the CONNECT / command, the Windows NT identity of both users must match or the second user can receive the same database roles and privileges as the first user. This is a security concern if the second user possesses only the CONNECT and RESOURCE database roles, but accidently receives the DBA database role associated with the first user. For this situation, setting this parameter to OS_AUTH_LOGIN ensures that Windows NT security checking is performed. Furthermore, if OS_ROLES is set to true in the database server, the roles of the operating system user are associated with a connection regardless of whether CONNECT / or CONNECT *username/password* is performed. To enable Windows NT security checking in this case, set this parameter to ALWAYS. Windows NT security checking is a resource-intensive operation. There is always a cost associated with Windows NT verifying the operating system credentials prior to reusing a connection. For performance reasons, it is advisable to set this parameter to NEVER. However, if you set OS_ROLES to true or use operating system-authenticated connections, ensure that you set this parameter accordingly.	There are three possible values: ■ ALWAYS Windows NT security checking is always performed. This setting is the most secure, because it does not permit a second user to accidently receive the database roles and privileges of the first user. ■ OS_AUTH_LOGIN (default) Windows NT security checking is only done if the username and password are NULL. This is the default value. ■ NEVER No Windows NT security checking is performed. This setting is the least resource intensive of the three. Use this setting if you are not setting OS_ROLES to true or not using operating system-authenticated connections.

Increasing the Transaction Timeout Parameter on Windows NT

If transaction requests are timing out before completing, the transaction timeout parameter may be set too low. Increase the transaction timeout parameter to ensure that transactions have enough time to complete.

To increase the transaction timeout parameter on Windows NT:

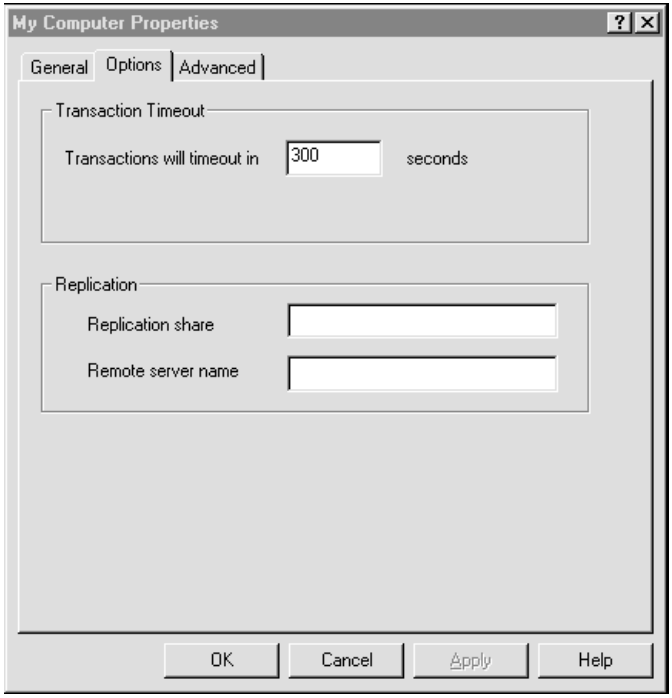
1. Go to the computer on which Microsoft Transaction Server is installed.
2. Choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Transaction Server > Transaction Server Explorer.

   The **Microsoft Management Console** appears.
3. Double-click Console Root in the Microsoft Management Console Explorer window.
4. Double-click Microsoft Transaction Server.
5. Double-click Computers.
6. Right-click My Computer.

   A menu appears with several options.
7. Choose Properties.

   The My Computer Properties dialog box appears.
8. Choose the Options tab.

9. Enter a value in the Transaction Timeout field and click OK:



The transaction timeout value is increased. For most environments, 60 seconds may be enough. However, if the transaction is competing with numerous concurrent transactions, this value may be too low.

See Also: Appendix A, "Using Oracle Services for Microsoft Transaction Server on Windows 2000" for information on increasing the transaction timeout parameter on Windows 2000

Changing Initialization Parameter File Settings

You may need to modify several initialization parameters to use the database server with Microsoft Transaction Server. The values to which to set these parameters are based upon the database workload environment.

To verify initialization parameter file values:

1. Ensure that you have SYSDBA privileges.
2. Go to the computer on which the database server is installed.
3. Start SQL*Plus:

   ```
   C:\> sqlplus/NOLOG
   ```

4. Connect to the database as SYSDBA:

   ```
   ENTER USER-NAME: / AS SYSDBA
   ```

5. Check the value for the SESSIONS parameter:

   ```
   SQL> SHOW PARAMETER SESSIONS
   ```

6. Check the value for the PROCESSES parameter:

   ```
   SQL> SHOW PARAMETER PROCESSES
   ```

   The current settings for both parameters are typically appropriate for running the Microsoft application demo. For creating and deploying COM-based applications, the values to which to set these parameters depend upon the anticipated workload for the database environment. For example, if you anticipate 100 concurrent connections to the database server, consider setting both values to 200 to account for any system overload. Ensure that you do not set these parameters too high, because they are resource-intensive. See *Oracle9i Database Reference* for information about these parameters.

To set initialization parameters:

1. Review the following table for the location of the initialization parameter file for the database server release:

If Using An...	Go to...
8.1.*x* or higher database	*ORACLE_BASE*\admin*DB_NAME*\pfile\init.ora
8.0.6 database	*ORACLE_HOME*\database\init*sid*.ora

2. Open the initialization parameter file.
3. Set the following initialization parameters to at least these values:
   - SESSIONS = 200 (or larger if anticipating heavier loads)
   - PROCESSES = 200 (or larger if anticipating heavier loads)
4. Shut down the database server:

   ```
   SQL> SHUTDOWN
   ```

5. Restart the database server:

   ```
   SQL> STARTUP
   ```

6. Exit SQL*Plus:

   ```
   SQL> EXIT
   ```

Starting MS DTC

The **Microsoft Distributed Transaction Coordinator (MS DTC)** must be running to enable communication with Oracle Services for Microsoft Transaction Server.

To start MS DTC:

1. Go to the computer on which Microsoft Transaction Server is installed.
2. Choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Transaction Server > Transaction Server Explorer.

   The Microsoft Management Console appears.
3. Double-click Console Root in the Microsoft Management Console Explorer window.
4. Double-click Microsoft Transaction Server.
5. Double-click Computers.
6. Right-click My Computer.

   A menu appears with several options.
7. Choose Start MS DTC.

   MS DTC starts.

7

Troubleshooting Oracle Services for Microsoft Transaction Server

This chapter provides **Oracle Services for Microsoft Transaction Server** troubleshooting information.

This chapter contains these topics:

Tracking Oracle Services for Microsoft Transaction Server Performance

Trace files record information about Oracle Services for Microsoft Transaction Server performance. This information includes:

- Any errors
- Enlistment requests and outcomes
- Prepare, commit, and abort requests and their outcomes

There are two registry parameters that handle tracing within `oramts.dll`. `oramts.dll` performs the following:

- Implements the API for integrating the Oracle database server with Microsoft Transaction Server
- Works as a resource dispenser to provide pooled **Oracle Call Interface (OCI)** connections
- Enables clients with nonpooled OCI connections to enlist in transactions started by **Microsoft Distributed Transaction Coordinator (MS DTC)**
- Communicates with Oracle Services for Microsoft Transaction Server to enlist the database server in MS DTC-started transactions

Table 7–1 describes the registry parameters for handling tracing. If not previously set, both parameters are automatically set in `\\HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOME`*`ID`* during Oracle Services for Microsoft Transaction Server installation. Use the registry parameters instead of setting these parameters as environment variables. Setting environment variables requires you to reboot your computer for the changes to occur.

Table 7–1 Trace Registry Parameter

Parameter	Description	Datatype	Default Value
`ORAMTS_CP_TRACE_LEVEL`	Traces the resource dispenser layer within `oramts.dll`. The trace filename uses the following format: `oramtscp`*`pid`*`.trc` where *`pid`* is the identification number of the process. New trace information is always added to the bottom of the file.	`REG_SZ`	`0`
`ORAMTS_CP_TRACE_DIR`	Sets the output directory in which trace files are generated.	`REG_SZ`	*`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\oramts\trace`

Table 7–2 shows the range of ORAMTS_CP_TRACE_LEVEL trace values. Set ORAMTS_CP_TRACE_LEVEL to a value other than 0 only when tracing is necessary.

Table 7–2 Trace Registry Parameter Value

Level	Description
0	Disables tracing[1]
1	Traces errors only
2	Traces important events in addition to errors
4	Traces function entry / exit, important events, and errors
5	Traces reference counting function and constructor / destructor entry / exit

[1] If the registry parameter is not set in the registry or as an environment variable, then tracing is disabled (the same as setting the level to 0). Note also that Level 3 is not currently supported. If you set this parameter to 3, level 2 tracing is instead enabled.

Note: The **Oracle MTS Recovery Service** also generates trace file output in the *ORACLE_BASE**ORACLE_HOME*\oramts\trace directory.

Correcting Windows NT Explorer Problems

If you experience Windows NT Explorer crashes or other unexpected Windows NT problems when using Microsoft Transaction Server with a database server, install the Windows NT 4.0 Service Pack 5 or greater (available from Microsoft).

Correcting Oracle Net Changes that Impact Connection Pooling

The connection pool provided by the ORAMTS layer (that is, `oramts.dll`) uses a connection's **net service name** to identify pooled connections for an application. If changes are made to the net service name, and there are currently pooled connections, the application using the connection pool must be stopped and restarted. These changes can include altering the host or the database system identifier (SID) for the net service name in the `tnsnames.ora` file.

These changes ensure that all currently pooled connections corresponding to the old net service name are destroyed and any new pooled connections use the changes made to the net service name. This includes any application hosting **Microsoft Transaction Server** components.

To empty connection pools:

1. Perform the instructions listed in the following table:

If the Application Is an...	Then...
Out-of-process Microsoft Transaction Server component (server package)	Run the following application: `C:\> mtxstop` This empties the connection pools.
In-process Microsoft Transaction Server component (library package)	Terminate the application, which also empties the connection pool.

Frequently Asked Questions About Oracle Services for Microsoft Transaction Server

This section presents answers to common questions.

Question: How do I design an application when I have multiple database servers?

Answer: Oracle clients can establish connections to a database server in two ways:

- Typical Oracle clients establish connections to a database server using a dedicated server configuration. In a dedicated server configuration, one client corresponds to one Oracle server process.
- For scalability under heavy loads, Oracle clients have the option of using a shared server configuration. In a shared server configuration, a single Oracle server process can be shared by more than one client connection.

Microsoft Transaction Server communicates with the database server through distributed transactions. In a dedicated server configuration, you cannot use distributed updates (**data manipulation language** statements across database links) from other database servers. However, if the original connection to the database server is established using shared server configurations, the distributed updates from other database servers succeed.

To use data manipulation language statements in shared server configurations, set the following parameter in the `tnsnames.ora` file:

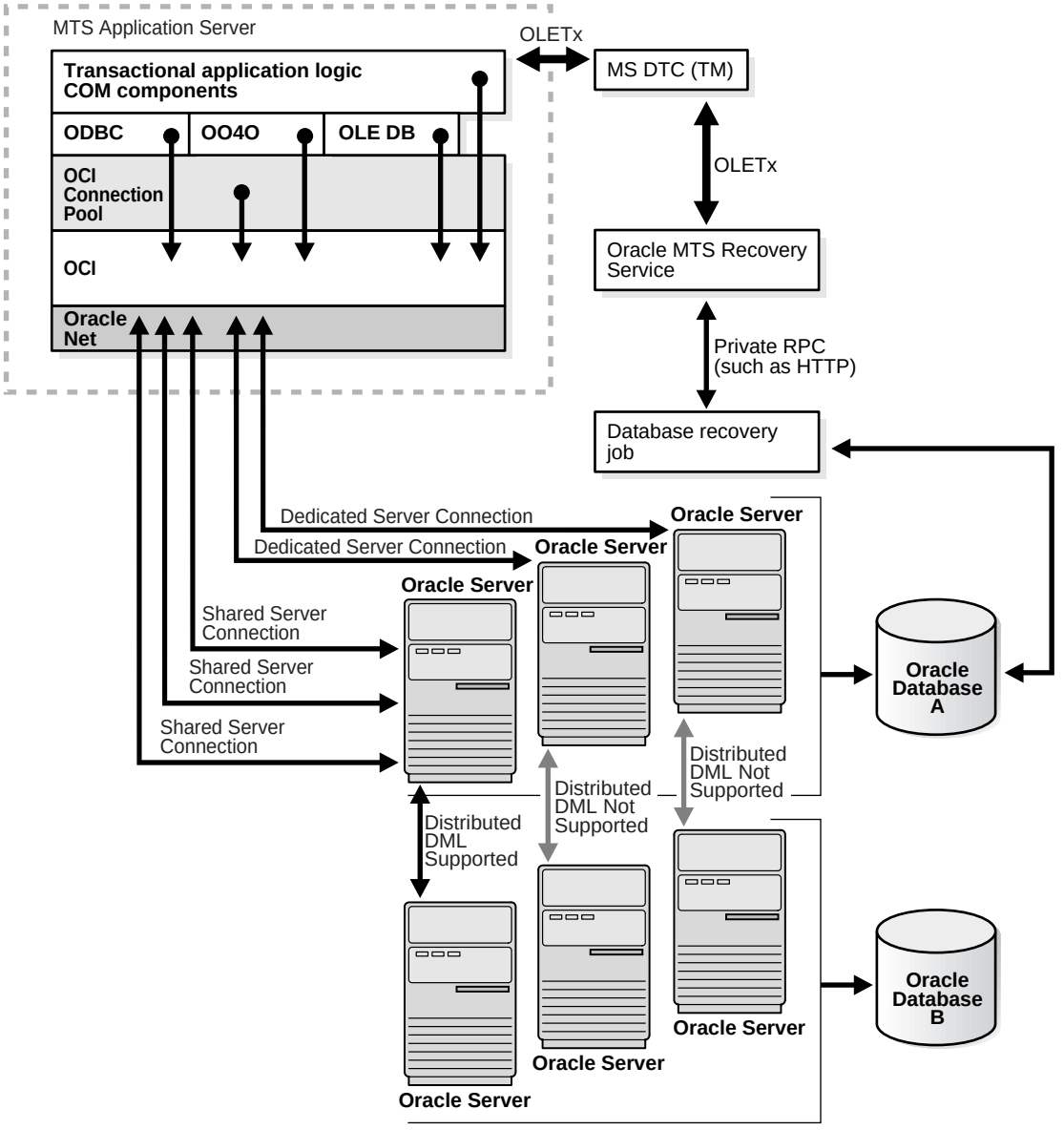
```
SERVER=dedicated
```

This forces the Oracle Net listener to provide a dedicated connection.

See Also: *Oracle9i Net Services Administrator's Guide*

Figure 7–1 shows this process.

Figure 7–1 Distributed DML statements from MTS applications



Question: What are the differences between Oracle Net connection pooling, OCI connection pooling, and Microsoft Transaction Server connection pooling?

Answer: Oracle Net connection pooling is a server-side feature that is implemented only if the database server is configured for shared server support. Oracle Net connection pooling enables you to minimize the number of physical network connections to a shared server. This is achieved by sharing a dispatcher's set of connections among multiple client processes.

Microsoft Transaction Server provides a resource pooling infrastructure that enables certain resources to be pooled, such as memory and database connections. The OCI connection pooling layer works with Microsoft Transaction Server resource pooling to provide pooled Oracle client/server sessions. The OCI connection pooling layer also caches Oracle Net connections to reduce client/server session setup time.

Question: What are in-doubt transactions?

Answer: Oracle uses distributed transactions in the following configurations:

- Distributed database configurations (for example, distributed updates using database links)
- External transaction monitors (for example, Tuxedo, MS DTC) for coordinating transaction outcome

The two-phase commit protocol completes these transactions. During phase one, the transaction monitor (TM) requests the various resource managers involved in the TM's transaction to prepare the underlying distributed transactions. In phase two, the TM determines whether it commits or aborts the transaction, and requests the resource managers to commit or abort the underlying transaction. If a resource manager fails to receive the phase two notification, the underlying distributed transaction becomes in-doubt.

To integrate Oracle with Microsoft Transaction Server, distributed transactions are used in the database. Distributed transactions correspond to transactions coordinated by the MS DTC. A distributed transaction can become in-doubt when the transaction cannot commit or abort (phase two of the two-phase commit). This occurs when the Microsoft Transaction Server application server process, database server, or network fails.

See Also: Chapter 3, "Managing Recovery Scenarios"

Dropping the Microsoft Transaction Server Administrative User Account

The Microsoft Transaction Server administrative user account is created by running the `oramtsadmin.sql` script with Oracle9*i* release 9.0.1. If you later change the database server with which Microsoft Transaction Server is coordinating transactions, you can drop the administrative user account schema from the previous database server.

To drop the Microsoft Transaction Server administrative user account:

1. Start SQL*Plus:

   ```
   c:\> sqlplus/NOLOG
   ```

2. Connect to the database server as SYSDBA:

   ```
   ENTER USER-NAME: / AS SYSDBA
   ```

3. Enter the following command to drop administrative user account schema:

   ```
   SQL> DROP USER mtsadmin_username CASCADE;
   ```

   where *mtsadmin_username* is the Microsoft Transaction Server administrative user account (default is `mtssys`).

4. See Chapter 3, "Managing Recovery Scenarios" for information on creating the Microsoft Transaction Server administrative user account for the new database server.

A

Using Oracle Services for Microsoft Transaction Server on Windows 2000

This appendix describes differences between using **Oracle Services for Microsoft Transaction Server** on Windows NT and Windows 2000.

This appendix contains these topics:

- Microsoft Transaction Server Differences on Windows NT and Windows 2000
- Increasing the Transaction Timeout Parameter on Windows 2000

Note: Oracle Services for Microsoft Transaction Server configuration is the same on Windows 2000 and Windows NT.

Microsoft Transaction Server Differences on Windows NT and Windows 2000

Table A–1 describes the differences between using **Microsoft Transaction Server** on Windows NT and Windows 2000.

Table A–1 Microsoft Transaction Server Differences on Windows NT and Windows 2000

Feature	On Windows NT	On Windows 2000
Microsoft Transaction Server	Microsoft Transaction Server is an add-on.	The functionality of Microsoft Transaction Server is integrated within the operating system as a COM+ server (COM+ transactions).
Microsoft Management Console	You must obtain the Microsoft Management Console from Microsoft Corporation.	Microsoft Management Console is automatically included.
Component object model	COM is the name.	COM+ is the name.
Installing and configuring transactional applications	Choose Start > Programs > Windows NT 4.0 Option Pack > Microsoft Transaction Server > Transaction Server Explorer	Choose Start > Programs > Administrative Tools > Component Services
Transaction Coordinator	Microsoft Distributed Transaction Coordinator (MS DTC) is the name.	Distributed Transaction Coordinator (DTC) is the name.
Microsoft Application Demo	The Microsoft application demo, a Visual C/C++ bank sample application, is available with Microsoft Transaction Server.	The Microsoft application demo is not shipped.

Increasing the Transaction Timeout Parameter on Windows 2000

If transaction requests are timing out before completing, the transaction timeout parameter may be set too low. Increase the transaction timeout parameter to ensure that transactions have enough time to complete.

To increase the transaction timeout parameter on Windows 2000:

1. Go to the Windows 2000 computer on which Microsoft Transaction Server is installed.
2. Choose Start > Programs > Administrative Tools > Component Services.

   The Microsoft Management Console appears.
3. Double-click Console Root in the Microsoft Management Console Explorer window.
4. Double-click Microsoft Transaction Server.
5. Double-click Computers.
6. Right-click My Computer.

   A menu appears with several options.
7. Choose Properties.

   The My Computer Properties dialog box appears.
8. Choose the Options tab.
9. Enter a value in the Transaction Timeout field and click OK.

   The transaction timeout value is increased. For most environments, `60` seconds may be enough. However, if the transaction is competing with numerous concurrent transactions, this value may be too low.

Glossary

Atomicity, Consistency, Isolation, and Durability (ACID)

ACID consists of the four primary attributes provided to any transaction by a transaction manager (also called a transaction monitor).

component object model (COM)

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

distributed component object model (DCOM)

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

data manipulation language

The category of SQL statements that query and update database data. Common DML statements are `SELECT`, `INSERT`, `UPDATE`, and `DELETE`.

JOB_QUEUE_PROCESSES

This initialization parameter specifies the number of job queue processes started in an instance. This parameter must be set to at least `1` to run job queue processes.

listener.ora

A listener configuration file that identifies the following for a listener:

- Unique name
- Protocol addresses on which it accepts connection requests
- Services for which it is listening

Microsoft Distributed Transaction Coordinator (MS DTC)

The focal point of the transaction process is a component of Microsoft Transaction Server called Microsoft Distributed Transaction Coordinator (MS DTC).

Microsoft application demo

An Oracle Call Interface (OCI) implementation of the Visual C++ Sample Bank package that ships with Microsoft Transaction Server on Windows NT.

Microsoft Management Console

An application that serves as a host for administrative tools called snap-ins. By itself, Microsoft Management Console does not provide any functionality.

Microsoft Transaction Server

A COM-based transaction processing system that runs on an Internet or network server.

mtssys

The default Microsoft Transaction Server administrator username. In releases prior to Oracle9*i* release 9.0, this was the username for the Oracle Service for MTS.

net service name

The name used by clients to identify an Oracle Net server and the specific system identifier (SID) or database for the Oracle Net connection. A net service name is mapped to a port number and protocol. A net service name is also known as a connect string, database alias, host string, or service name.

This also identifies the specific SID or database server to which the connection is attaching, and not just the Oracle Net server.

ORAMTS_ORADB

Database alias for connecting the Oracle Service for MTS to the Oracle database (no longer supported with Oracle Services for Microsoft Transaction Service, release 9.0).

Oracle Call Interface (OCI)

An application programming interface that enables you to manipulate data and schemas in a database server. You compile and link an OCI program in the same way that you compile and link a nondatabase application. There is no requirement for a separate preprocessing or precompilation step.

Oracle Fail Safe

Ensures that if a failure occurs on one cluster node, then the database servers and applications running on that node fail over (move) automatically and quickly to a surviving node.

Oracle Manager for MTS Services snap-in

This Microsoft Management Console snap-in is no longer available or required with Oracle Services for Microsoft Transaction Server, release 9.0.

Oracle MTS Recovery Service

The Oracle MTS Recovery Service resolves in-doubt transactions on the computer that started the failed transaction. A scheduled recovery job for each Microsoft Transaction Server-enabled database server lets the Oracle MTS Recovery Service resolve in-doubt transactions.

Oracle Objects for OLE (OO4O)

A custom control (OCX or ActiveX) combined with an object linking and embedding (OLE) in-process server that lets you plug native Oracle database functionality into your Windows applications.

Oracle Open Database Connectivity (ODBC) Driver

Oracle ODBC Driver provides a standard interface that allows one application to access many different data sources. The application's source code does not have to be recompiled for each data source. A database driver links the application to a specific data source. A database driver is a dynamic link library that an application can invoke on demand to gain access to a particular data source. Therefore, the application can access any data source for which a database driver exists.

Oracle Provider for OLE DB

Interfaces that offer high performance and efficient access to Oracle data by applications, compilers, and other database components.

Oracle Service for MTS

This Windows NT service is no longer available or required with Oracle Services for Microsoft Transaction Server, release 9.0.

Oracle Services for Microsoft Transaction Server

A component that provides full integration of Oracle database releases 8.0.6 and greater with Microsoft Transaction Server. This component enables you to develop and deploy COM-based applications using Microsoft Transaction Server.

Optimal Flexible Architecture (OFA)

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

resource manager (RM)

Microsoft Transaction Server enlists the database server to act as a resource manager (RM) in the transaction process.

SYSDBA

A special database administration role that contains all system privileges with the ADMIN OPTION and the SYSOPER system privilege. SYSDBA also permits CREATE DATABASE actions and time-based recovery.

SYSOPER

A special database administration role that permits a database administrator to perform STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, and RECOVER, and includes the RESTRICTED SESSION privilege.

tnsnames.ora

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

transaction identifiers (XIDs)

Identifies the client computer from which a transaction originated.

Index

A

C

M

N

O

P

R

S

T

U

V

W