Oracle® Services for Microsoft Transaction Server

Developer's Guide

10*g* Release 1 (10.1) for Windows

Part No. B10114-01

December 2003

ORACLE®

Oracle Services for Microsoft Transaction Server Developer's Guide 10*g* Release 1 (10.1) for Windows

Part No. B10114-01

Primary Author: Janelle Simmons

Contributing Author: Mark Kennedy

Contributors: Valarie Moore, Vivek Raja, Eric Wang

Contents

Send Us Your Comments

Oracle Services for Microsoft Transaction Server Developer's Guide, 10*g* Release 1 (10.1) for Windows

Part No. B10114-01

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

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

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

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

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

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

Preface

This document is the primary source of introduction, installation, configuration, usage, and administration information for using Oracle Services for **Microsoft Transaction Server** that apply to operating systems.

This document describes the features of Oracle Database for Windows software that apply to the Windows NT Server, Windows 2000, Windows XP, and Windows Server 2003 operating systems.

This preface contains these topics:

Audience

Oracle Services for Microsoft Transaction Server is intended for anyone who performs the following tasks:

- Uses **component object model (COM)** components with Microsoft Transaction Server
- Registers COM components as transactional and has Microsoft Transaction Server control the transaction
- Uses client-side connection pooling in Microsoft Transaction Server
- Uses .NET applications with Oracle Services for Microsoft Transaction Server to access Oracle databases.

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.

Chapter 2, "Installing and Migrating Oracle Products"

This chapter describes installation and migration requirements for Microsoft Transaction Server and database 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"

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

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 Operating Systems"

This appendix describes differences between using Oracle Services for Microsoft Transaction Server on Windows NT Server, Windows 2000, Windows XP, and Windows Server 2003.

Glossary

Related Documentation

For more information, see these Oracle resources:

- *Oracle Database Reference*
- *Oracle Provider for OLE DB Developer's Guide*
- *Oracle Objects for OLE Developer's Guide*
- *Oracle Data Provider for .NET Developer's Guide*
- *Oracle Net Services Administrator's Guide*
- *Oracle Database Platform Guide for Windows*

For information about Oracle error messages, see *Oracle Database Error Messages*. Oracle error message documentation is available only in HTML. If you only have access to the Oracle Documentation CD, you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

Many of the examples in this book use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

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

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

`http://otn.oracle.com/membership/`

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

`http://otn.oracle.com/documentation/`

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.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.

Convention	Meaning	Example
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.
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 italic monospace (fixed-width) font	Lowercase italic monospace 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]`

`...`	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.	`SQL> SELECT NAME FROM V$DATAFILE;`
		`NAME`
		`------------------------------------`
		`/fsl/dbs/tbs_01.dbf`
		`/fs1/dbs/tbs_02.dbf`
		`.`
		`.`
		`.`
		`/fsl/dbs/tbs_09.dbf`
		`9 rows selected.`

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;`

Convention	Meaning	Example

`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.	To start the Database Configuration Assistant, choose **Start** > **Programs** > **Oracle** - ***HOME_ NAME*** > **Configuration and Migration Tools** > **Database Configuration Assistant**.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\\\, then 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 the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`

Convention	Meaning	Example

Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to 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`
`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. For Windows NT, the default location was `C:\orant`. 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 the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\ora`*`nn`*, where *`nn`* is the latest release number. The Oracle home directory is located directly under *`ORACLE_BASE`*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Platform Guide 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

1 Using Microsoft Transaction Server with Oracle

This chapter describes **Microsoft Transaction Server** and Oracle database 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. Microsoft Transaction Server functionality was later implemented in COM+ and Enterprise Services. Microsoft Transaction Server, COM+, and Enterprise Services are supported by Oracle Services for Microsoft Transaction Server.

See Also: Microsoft documentation for additional information about Microsoft Transaction Server

Oracle Services for Microsoft Transaction Server Support for Serializable Transactions

Oracle Services for Microsoft Transaction Server now supports distributed transactions set to a serializable isolation level.

Microsoft Transaction Server and Oracle Integration Overview

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

- Register the COM component or .NET application 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 helps in the recovery of in-doubt transactions left in Oracle databases that originated from this computer. On each connected database perform the following:

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

This enables the database 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**

Access the Oracle database using any .Net application using **Oracle Data Provider for .NET (ODP.NET)**.

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

Getting Started with Microsoft Transaction Server and Oracle

You are now ready to use Microsoft Transaction Server with a database. 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 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 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" for instructions on using OCI, OO4O, Oracle ODBC Driver, or Oracle Provider for OLE DB with COM-based applications.
Learn about using Microsoft Transaction Server on the different Windows operating systems.	Appendix A, "Using Oracle Services for Microsoft Transaction Server on Windows Operating Systems"

2 Installing and Migrating Oracle Products

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

This chapter contains these topics:

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 *Oracle Database Installation Guide for Windows* for instructions on installing the required Oracle products. Key guidelines to understand are:

- At least 256 MB of RAM are required if Oracle database, Microsoft Transaction Server, and **Oracle Services for Microsoft Transaction Server (OraMTS)** are installed on the same computer.
- Only one **Oracle MTS Recovery Service** exists on each computer.
- For **Oracle Data Provider for .NET (ODP.NET)** 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.

Table 2–1 Installation Requirements

On This Computer...	Oracle Products	Non-Oracle Products
Windows computer where Microsoft Transaction Server is installed	▪ Oracle Services For Microsoft Transaction Server ▪ Oracle Net Services for the client ▪ **Oracle Objects for OLE (OO4O)** ▪ Oracle Open Database Connectivity (ODBC) Driver ▪ Oracle Provider for OLE DB ▪ Oracle Call Interface (OCI) ▪ Oracle Data Provider for .NET (ODP.NET) ▪ Oracle Net Manager ▪ ORACLE_LOADER (data access driver) ▪ SQL*Plus **Note**: Oracle Net Services is automatically installed with Oracle Services for Microsoft Transaction Server.	▪ Microsoft Transaction Server version 2.0 or later versions ▪ Windows operating system ▪ Fully-functioning networking protocol software **Note:** COM+ or Enterprise Services are later versions of Microsoft Transaction Server.
Computer where the Oracle database is installed	▪ Oracle Server (the Oracle database) Oracle Net Services for the server ▪ SQL*Plus	▪ Fully-functioning networking protocol software

Note: OO4O, Oracle ODBC Driver, ODP.NET, Oracle Provider for OLE DB, and OCI are only required if you are building or using components with which they are required.

See Also: *Oracle 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.

Upgrading from a Previous Oracle Services For Microsoft Transaction Server Installation

Before uninstalling 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 Upgrade Requirements

Requirement	See
Delete the Oracle Service for MTS using the Oracle Manager for MTS Services snap-in. **Note:** If you have already deleted the database, 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 Oracle Manager for MTS Services" on page 2-5 "Using the Registry to Manually Delete the Oracle Service for MTS" on page 2-10 if you have already deleted the database
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 2-9
Complete the following installation tasks: ▪ Uninstall Oracle Services For Microsoft Transaction Server from the Windows computer where Microsoft Transaction Server is installed. ▪ Install release 10.1 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. ▪ Install Oracle Services For Microsoft Transaction Server release 10.1 into a single Oracle home. The Oracle MTS Recovery Service is also automatically installed.	*Oracle Database Installation Guide for Windows*
Complete the following postinstallation tasks: ▪ Create the Microsoft Transaction Server administrator user account. ▪ Schedule Microsoft Transaction Server transaction recovery jobs for all databases that participate in Microsoft Transaction Server transactions.	Chapter 3, "Managing Recovery Scenarios"

Deleting an Oracle Service for MTS with Oracle Manager for MTS Services

You must use 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 Delete button on the keyboard) causes data inconsistencies in the database. These inconsistencies require the database administrator to manually commit or terminate 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.

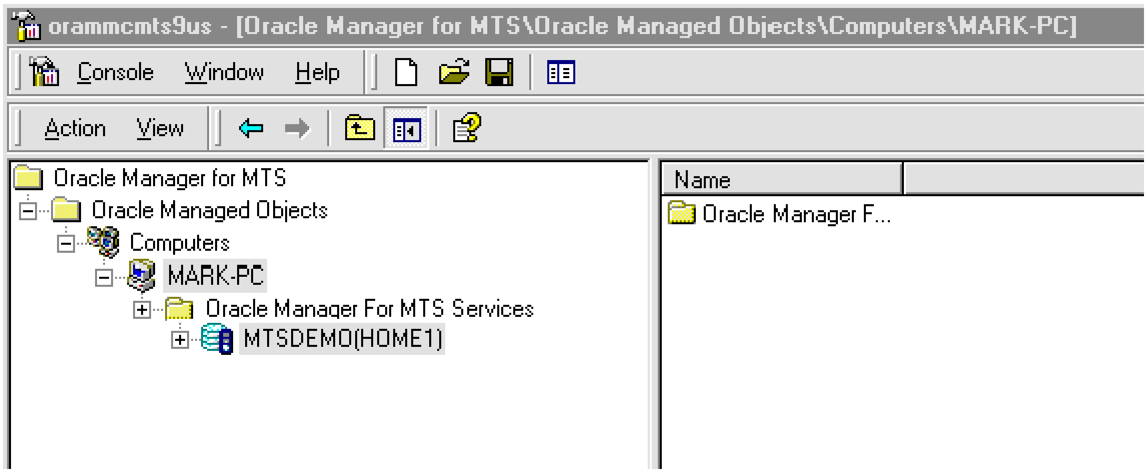
- Task 1: Stopping the Oracle Service for MTS
- Task 2: Stopping and Restart the Database
- Task 3: Restarting the Oracle Service for MTS
- Task 4: Monitoring the Oracle Service for MTS Trace Files
- Task 5: Deleting Oracle Service for MTS Table Information
- Task 6: Deleting the Oracle Service for MTS

Task 1: Stopping 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: Stopping and Restart the Database

To stop and restart the database:

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

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

3. Connect to the database as `SYSDBA`:

   ```
   SQL> CONNECT / AS SYSDBA
   ```

4. Shut down the Oracle database:

   ```
   SQL> SHUTDOWN
   ```

5. Restart the Oracle database:

   ```
   SQL> STARTUP
   ```

6. Exit SQL*Plus:

   ```
   SQL> EXIT
   ```

Task 3: Restarting 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: Monitoring 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: Deleting Oracle Service for MTS Table Information

To delete Oracle Service for MTS table information:

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

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

3. Connect to the database as `SYSDBA`:

   ```
   SQL> CONNECT / 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: Deleting 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 2-9
The Oracle Service for MTS was not deleted.	"Using the Registry to Manually Delete the Oracle Service for MTS" on page 2-10

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`:

```
SQL> CONNECT / 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
```

See Also: *Oracle 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 Oracle database 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 Oracle Manager for MTS Services" on page 2-5 were unsuccessful
- You have already deleted the Oracle database 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 Oracle database using the information in the registry.	Ensure that the Oracle database and its listener are started. Use SQL*Plus or a different tool to verify that the Oracle database accepts new connections.

Table 2–3 (Cont.) Oracle Service for MTS Deletion or Modification Failures

Scenario	Solution
The information in the Oracle database does not match the information in the registry.	The Oracle Manager for MTS Services snap-in is connecting to a different Oracle database 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.
Oracle Manager for MTS Services snap-in cannot delete the service information stored in Oracle database.	Oracle database 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 Deleting Oracle Service for MTS with the Registry

To manually delete Oracle Service for MTS with the registry:

1. Start the registry from the command prompt:

   ```
   C:\> regedt32
   ```

   The Registry Editor window appears.

2. Select the `HKEY_LOCAL_MACHINE`.

   Go to `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.

Parameter	This Parameter Contains...
ORAMTS_SUPWD	The password for the Oracle Service for MTS username (encrypted in the registry).
ORAMTS_OCI_OBJ_MODE	Initializes OCI in object mode or threaded mode
ORAMTS_ORADB	The **net service name** for the Oracle Service for MTS to use in connecting to the Oracle database.

3. Start SQL*Plus:

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

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

```
CONNECT as 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 Oracle database 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 all values do not match, the Oracle database is not the same one to which the Oracle Service for MTS connects. If you continue, Oracle Service for MTS installation on the database fails. This can leave the database in an inconsistent state that requires database administrator intervention to correct. Because of mismatched `tnsnames.ora` files, SQL*Plus and Oracle Service for MTS did not connect to the same database.
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: Deleting the OracleMTSService*n* Service

To delete the `OracleMTSService`*`n`* service:

1. Restart the computer.
2. Click **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 **Oracle Service for MTS**.

   A menu appears with several options.
5. Click **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. In the `HKEY_LOCAL_MACHINE`, delete the following key:

   `\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 2-9.

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 environments after installing or migrating **Oracle Services for Microsoft Transaction Server (OraMTS)**. Review Table 3–1 to identify what you must configure.

Configuration is not required on the Windows computer where Microsoft Transaction Server is installed.

Perform the procedures listed in Table 3–1 on the computer where the Oracle database is installed.

Table 3–1 Configuration Requirements

Configuration Task	See
Run the `oramtsadmin.sql` script against the database to create the Microsoft Transaction Server administrative user account (the default username is `mtssys`). Schedule automatic transaction recovery.	"Scheduling Automatic Microsoft Transaction Server Transaction Recovery" on page 3-3
If you have an Oracle Fail Safe configuration, modify the registry values before or after running the `oramtsadmin.sql` script.	"Modifying Registry Values for Oracle Fail Safe Configurations" on page 3-10 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 permits the Oracle MTS Recovery Service to resolve in-doubt transactions.

The Oracle MTS Recovery Service resolves an in-doubt Microsoft Transaction Server transaction in the following order:

1. The DBMS recovery job detects an in-doubt MTS-related transaction.
2. The 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. The recovery service requests its MS DTC for transaction outcome.
4. The recovery service reports transaction outcome to the DBMS job process.
5. The DBMS recovery job commits or aborts the 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 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 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 by performing the following tasks:

- Task 1: Setting and Starting Up Database Job-Queue Processes
- Task 2: Creating and Scheduling Automatic Transaction Recovery

Task 1: Setting and Starting Up Database Job-Queue Processes

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

To set and start up job-queue processes:

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

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

4. Connect to the database as `SYSDBA`:

   ```
   SQL> CONNECT / AS SYSDBA
   ```

5. 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.

6. Shut down the Oracle database:

   ```
   SQL> SHUTDOWN
   ```

7. Restart the Oracle database:

   ```
   SQL> STARTUP
   ```

8. Exit SQL*Plus:

   ```
   SQL> EXIT
   ```

Task 2: Creating and Scheduling 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 Oracle database is installed.
3. Start SQL*Plus:

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

4. Connect to the database as `SYSDBA`:

   ```
   SQL> CONNECT / 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

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

Type

Procedure

Arguments

None

Returns

None

Description

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      : 5BAB6A6B55CD294AA20335839110829C01000000000901944700050
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

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 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.

Type

Procedure

Arguments

None

Returns

None

utl_oramts.forget_RMs Procedure

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

Type

Procedure

Arguments

None

Returns

None

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. This view consists of the following columns:

Formatid

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

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 terminated, and so on.

protocol

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

endpoint

This is the endpoint of the Windows 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, 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 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.

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, Windows XP, or Windows Server 2003.

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 (OraMTS)	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 to a different location. **3.** Copy the `vcacct.dll` file in *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\oramts\samples\account.vc\release` to the 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 created during installation.

See Also:

- "Ensuring the Database Includes the Proper Microsoft Application Demo Tables" for more information.
- Unlocking the Sample Tables in *SQL*Plus User's Guide and Reference* for more information on unlocking the sample schemas

Ensuring the Database Includes the Proper Microsoft Application Demo Tables

If the default database is not being used or the database 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 includes the proper tables:

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

If You Create the Database Through These Methods...	Then...
▪ Through the General Purpose, Transaction Processing, or Data Warehouse database configuration type of Oracle Universal Installer or Database Configuration Assistant ▪ Manually using a custom SQL script, and then explicitly run the `scott.sql` and `omtssamp.sql` scripts in that order against the database **Note**: If you run `omtssamp.sql` before running `scott.sql`, you receive numerous error messages. Ignore these messages. The product functions properly, but the sample application demo described in this chapter does not work.	The database includes the proper tables. Go to "Running the Microsoft Application Demo" on page 4-5.
▪ 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 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.

If the database 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:

   ```
   SQL> CONNECT username/password@net_service_name
   ```

   where `net_service_name` is the net service name that connects to the database. If you connect to the database 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:

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`.
2. Open the project `account.dsp` in the *ORACLE_BASE*`\`*ORACLE_HOME*`\oramts\SAMPLES\account.vc\Release` 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 numbers and the amount for the operation.
10. Click **Submit**.
11. Start SQL*Plus:

    ```
    C:\> sqlplus /NOLOG
    ```
12. Connect with the username `scott` and the password tiger (unless you changed it):

    ```
    SQL> CONNECT 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** 10*g* release 1 (10.1) with the Microsoft sample application.

> **See Also:** "Using Oracle ODBC Driver" on page 5-24 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** 10*g* release 1 (10.1) with the Microsoft sample application.

> **See Also:** *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

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

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 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

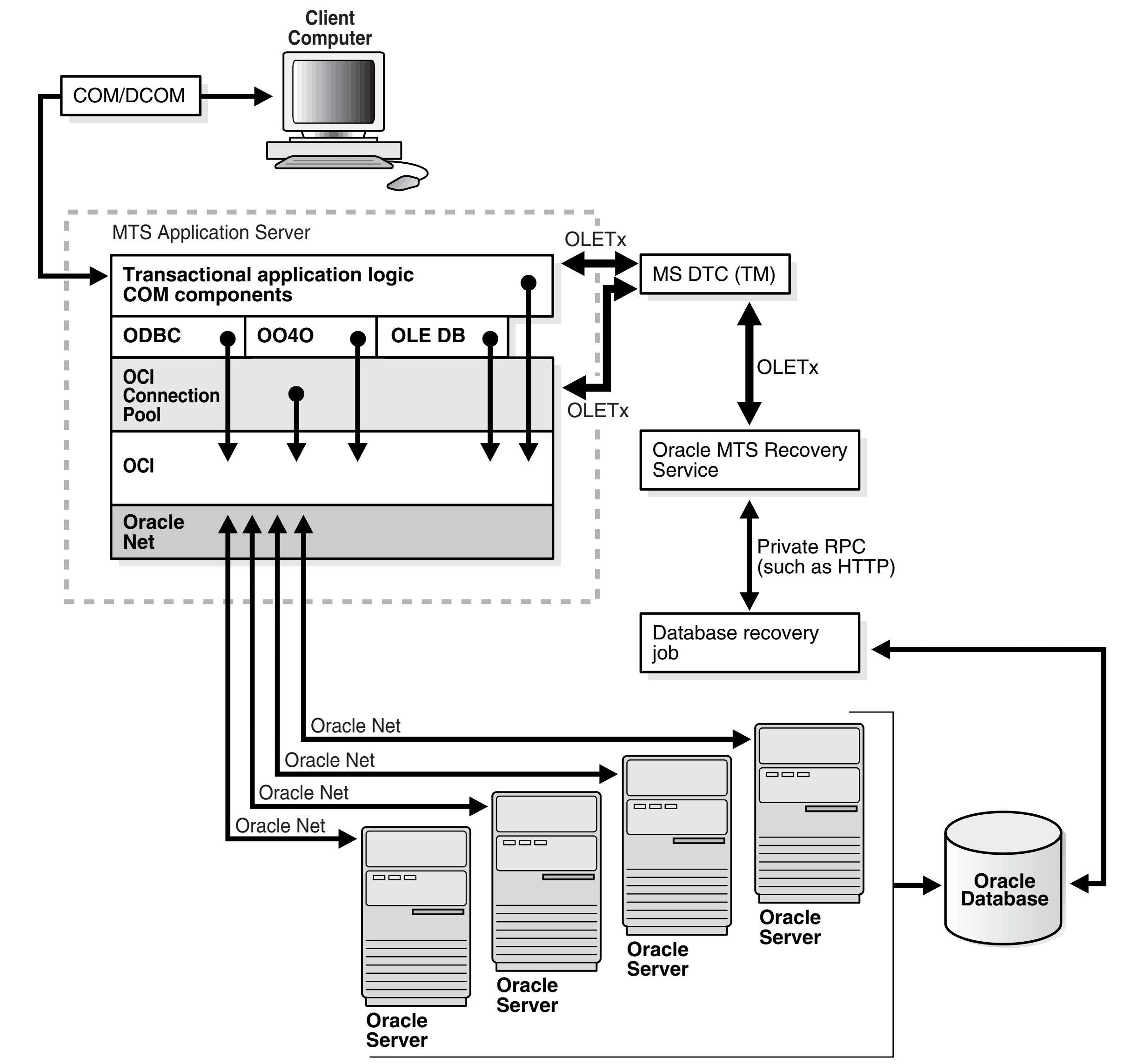


Table 5–1 Component Integration in a Transaction

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 Oracle database 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 terminate the changes to all RMs.
Oracle ODBC Driver, OO4O, Oracle Provider for OLE DB, and OCI	Obtain a service context to the Oracle database 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 (Oracle database) 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 terminates 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 terminates 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.
Oracle Database	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 how 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

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 terminated if the transaction fails. See Also: "Microsoft Transaction Server-Coordinated Component Transaction Overview" on page 5-6
Does not run in a Microsoft Transaction Server-coordinated transaction	The component runs in a Microsoft Transaction Server environment, but the databases 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 (such as Oracle Provider for OLE DB, Oracle ODBC Driver, OO4O, or others). See Also: "MS DTC-Coordinated Component Transaction Overview" on page 5-7

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, ODP.NET 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 terminate 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 terminate) 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 Oracle database. 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 Oracle database. Review the following sections for information about this integration:

- OCI and Microsoft Transaction Server Function Overview
- OraMTSSvcGet() Function
- OraMTSSvcRel() Function
- OraMTSSvcEnlist() Function
- OraMTSSvcEnlistEx() Function
- OraMTSEnlCtxGet() Function
- OraMTSEnlCtxRel() Function
- OraMTSJoinTxn() Function
- OraMTSTransTest() Function
- OraMTSOCIErrGet() Function

OCI and Microsoft Transaction Server Function Overview

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 Also: "OraMTSSvcGet() Function" on page 5-12 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 Also: "OraMTSJoinTxn() Function" on page 5-20 for more information

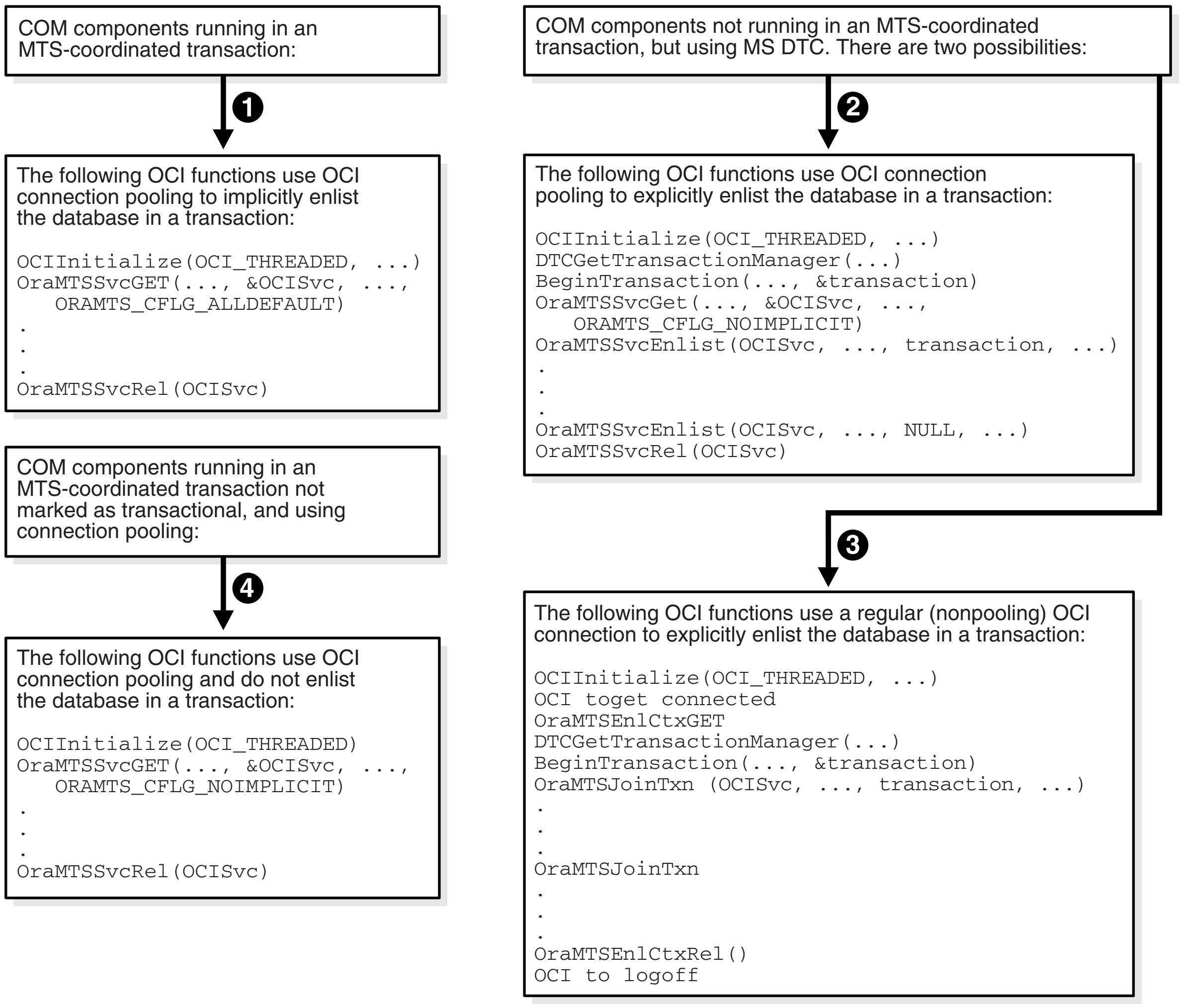
Ensure that for each process, you call `OCIInitialize` at least once before executing any other OCI calls. This initializes the OCI process environment. In addition, you must pass it the `OCI_THREADED` flag. If you are using Microsoft Internet Information Server (IIS) and the components are being called as in-process libraries, then `OCIInitialize` is already called for you. The registry key `ORAMTS_OCI_OBJ_MODE` has been added. Set the value to 1 to initialize OCI in Object mode; otherwise OCI will initialize in the threaded mode.

```
#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(
"hr","hr_password","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);
}
```

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 components running in an MTS-coordinated transaction:

The following OCI functions use OCI connection pooling to implicitly enlist the database in a transaction:

```
OCIInitialize(OCI_THREADED, ...)
OraMTSSvcGET(..., &OCISvc, ...,
   ORAMTS_CFLG_ALLDEFAULT)
.
.
.
OraMTSSvcRel(OCISvc)
```

COM components running in an MTS-coordinated transaction not marked as transactional, and using connection pooling:

The following OCI functions use OCI connection pooling and do not enlist the database in a transaction:

```
OCIInitialize(OCI_THREADED)
OraMTSSvcGET(..., &OCISvc, ...,
   ORAMTS_CFLG_NOIMPLICIT)
.
.
.
OraMTSSvcRel(OCISvc)
```

COM components not running in an MTS-coordinated transaction, but using MS DTC. There are two possibilities:

The following OCI functions use OCI connection pooling to explicitly enlist the database in a transaction:

```
OCIInitialize(OCI_THREADED, ...)
DTCGetTransactionManager(...)
BeginTransaction(..., &transaction)
OraMTSSvcGet(..., &OCISvc, ...,
   ORAMTS_CFLG_NOIMPLICIT)
OraMTSSvcEnlist(OCISvc, ..., transaction, ...)
.
.
.
OraMTSSvcEnlist(OCISvc, ..., NULL, ...)
OraMTSSvcRel(OCISvc)
```

The following OCI functions use a regular (nonpooling) OCI connection to explicitly enlist the database in a transaction:

```
OCIInitialize(OCI_THREADED, ...)
OCI toget connected
OraMTSEnlCtxGET
DTCGetTransactionManager(...)
BeginTransaction(..., &transaction)
OraMTSJoinTxn (OCISvc, ..., transaction, ...)
.
.
.
OraMTSJoinTxn
.
.
.
OraMTSEnlCtxRel()
OCI to logoff
```

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).

OraMTSSvcGet() Function

`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 5–5 describes the possible connections flags for the `ub4` datatype.

Table 5–4 OraMTSSvcGet() Parameters

Datatype	Parameter	Description
`text*`	`lpUname(IN)`	Username for connecting to the Oracle database
`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. Possible values are represented in Table 5–5.

Table 5–5 Possible Connection Flags for ub4 Datatype

Connection Flag	Description
`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.
`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 Oracle database 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.

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

To obtain a nonenlisted connection using the hr/*hr_password* account, call OraMTSSvcGet() as follows:

```
OraMTSSvcGet("hr", "hr_password", "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, hr/*hr_password*@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

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 5–6 describes the OraMTSSvcRel() parameters.

Table 5–6 OraMTSSvcRel() Parameters

Datatype	Parameter	Description
OCISveCtx*	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

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 5–7 describes the OraMTSSvcEnlist() parameters.

Table 5–7 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

`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 5–8 describes the `OraMTSSvcEnlistEx()` parameters.

Table 5–8 OraMTSSvcEnlistEx() 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.

Table 5–8 (Cont.) OraMTSSvcEnlistEx() Parameters

Datatype	Parameter	Description
`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.
`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

`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 5–9 describes the `OraMTSEnlCtxGet()` parameters.

Table 5–9 OraMTSEnlCtxGet() Parameters

Datatype	Parameter	Description
`text*`	`lpUname(IN)`	Username for connecting to the Oracle database
`text*`	`lpPsswd(IN)`	Password for connecting to the Oracle database
`text*`	`lpDbnam(IN)`	Net service name for connecting to a database
`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 terminate work.
7. Release the enlistment context by calling `OraMTSEnlCtxRel()`.
8. Release the nonpooled OCI connection and delete its associated OCI environment handle.

OraMTSEnlCtxRel() Function

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

Syntax

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

Parameters

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

Table 5–10 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

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

Syntax

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

Parameters

Table 5–11 describes the `OraMTSJoinTxn()` parameters.

Table 5–11 OraMTSJoinTxn() Parameters

Datatype	Parameter	Description
`void*`	`pCtxt(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

`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

`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 5–12 describes the `OraMTSOCIErrGet()` parameters.

Table 5–12 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("hr"
,
 "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 Oracle database. Specific topics discussed are:

- Setting the Connection Attribute
- Using Oracle ODBC Driver (recommended)
- Using Microsoft 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 10.1 or Microsoft 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 Also:** "Setting Up MTS to Access Oracle" in the Microsoft Transaction Server online Help for instructions.

Using Oracle ODBC Driver

The ODBC Driver Manager distributed with ODBC 3.0 is a Resource Dispenser that supports connection pooling. Oracle ODBC Driver release 10.1 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.

See Also: Microsoft Transaction Server SDK for information

Use the Oracle ODBC Driver 10.1 with:

- Applications you develop
- The sample banking application that Microsoft provides with Microsoft Transaction Server.

See Also: Chapter 4, "Running the Microsoft Application Demo" for more information

To configure Oracle ODBC Driver:

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

   a. Back up Microsoft `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 the Microsoft 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** in *HOME_NAME*.
7. Click **Advanced**.
8. Add the following information in the keywords and values field:

   ```
   SERVER=database_alias
   USERNAME=hr
   PASSWORD=hr_password
   ```

   The following table describes the keywords and values.

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

Note: Verify that the hr schema contains the account and receipt tables.

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 sample application	mtssamples.dsn (Microsoft 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 Oracle ODBC Driver

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

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

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

> **See Also:** "Setting Up MTS to Access Oracle" in the Microsoft Transaction Server online Help for instructions on enabling Microsoft Oracle ODBC Driver

After enabling Microsoft's Oracle ODBC Driver perform these additional steps:

To configure Microsoft's Oracle ODBC Driver:

1. Install Oracle Required Support Files (RSF) release 10.1 and SQL*Net 2.3 or later 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 10.1 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
   ```

Integration Overview

Oracle Services for Microsoft Transaction Server provides integration with OO4O, Oracle Provider for OLE DB, and Oracle Data Provider for .NET.

See Also:

- *Oracle Objects for OLE Developer's Guide* for additional information about using OO4O with Microsoft Transaction Server
- *Oracle Provider for OLE DB Developer's Guide* for information about using Oracle Provider for OLE DB with Microsoft Transaction Server
- *Oracle Data Provider for .NET Developer's Guide* for information about using Oracle Data Provider for .NET with Microsoft Transaction Server

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 databases, use database links and connect to one database from the COM component.

> **See Also:** "OCI Integration with Microsoft Transaction Server Overview" on page 5-8 for more information about using `ORAMTSSvcGet()`

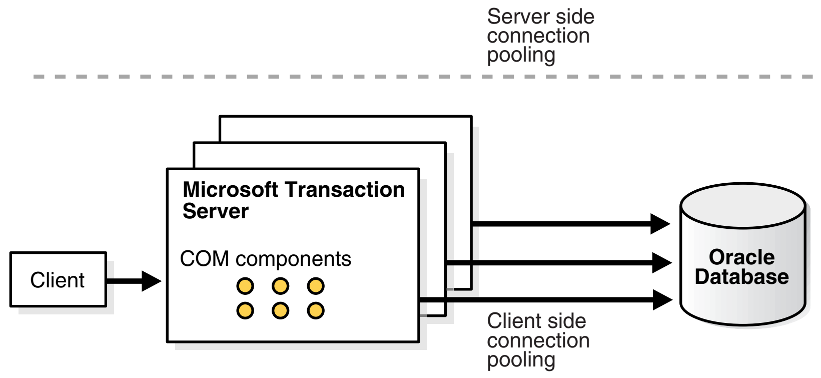
Managing Microsoft Transaction Server Connections

When a .NET or COM component ends a session with the Oracle database, 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 component attempting a new connection to the Oracle database.

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


Server side connection pooling
Microsoft Transaction Server
COM components
Client
Oracle Database
Client side connection pooling

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

Table 6–1 (Cont.) Connection Pool Registry Parameters

Client Side Parameter	Description	Default Value Entry
`ORAMTS_SESS_TXNTIMETOLIVE`	The portion of the connection associated with `ORAMTS_SESS_TXNTIMETOLIVE` specifies the time to live for Oracle client/server pooled connection established using OraMTSSvcGet(). This time is the time that a connection lives after being released by OraMTSSvcRel().	`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 than 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 (Cont.) 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 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 Oracle database (for example, with the `CONNECT /` command), there are certain database roles and privileges associated with their Windows username. When the user disconnects, the connection becomes idle and available in the pool. When another user enters the `CONNECT /` command, the Windows 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 security checking is performed. Furthermore, if `OS_ROLES` is set to `true` in the Oracle database, 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 security checking in this case, set this parameter to `ALWAYS`. Windows security checking is a resource-intensive operation. There is always a cost associated with Windows 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 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 security checking is only done if the username and password are `NULL`. This is the default value. ■ `NEVER` No Windows 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.

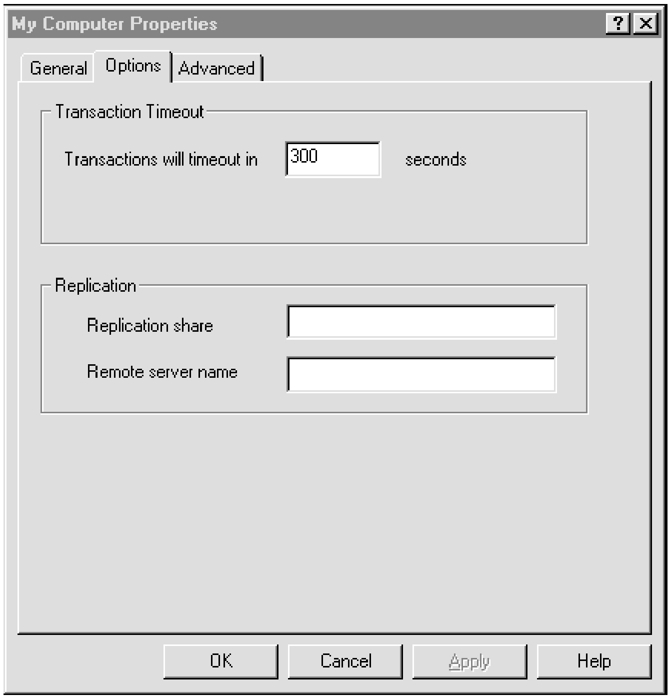
To increase the transaction timeout parameter on Windows NT:

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**.


My Computer Properties

Transaction Timeout

Transactions will timeout in 300 seconds

Replication

Replication share

Remote server name

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 Operating Systems" for information on increasing the transaction timeout parameter on Windows 2000

Changing Initialization Parameter Settings

You may need to modify several initialization parameters to use the Oracle database 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 Oracle database is installed.
3. Start SQL*Plus:

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

4. Connect to the database as `SYSDBA`:

   ```
   SQL> CONNECT / 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 .NET or 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 Oracle database, 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 Also:** *Oracle Database Reference* for information about these parameters.

To set initialization parameters:

1. 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)

2. Shut down the Oracle database:

   ```
   SQL> SHUTDOWN
   ```

3. Restart the Oracle database:

   ```
   SQL> STARTUP
   ```

4. Exit SQL*Plus:

   ```
   SQL> EXIT
   ```

Starting MSDTC

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** > **WindowsNT 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** MSDTC.

   MS DTC starts.

7 Troubleshooting Oracle Services for Microsoft Transaction Server

This chapter provides **Oracle Services for Microsoft Transaction Server (OraMTS)** 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 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 Oracle database 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 restart 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 **Note**: 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.
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

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

Correcting Windows Explorer Problems

If you are using Windows NT and experience Windows Explorer crashes or other unexpected Windows problems when using Microsoft Transaction Server with a database, install the Windows NT 4.0 Service Pack 6a 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:

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 databases?

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

- Typical Oracle clients establish connections to a database 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 through distributed transactions. In a dedicated server configuration, you cannot use distributed updates (**data manipulation language** statements across database links) from other databases. However, if the original connection to the database is established using shared server configurations, the distributed updates from other databases succeed.

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

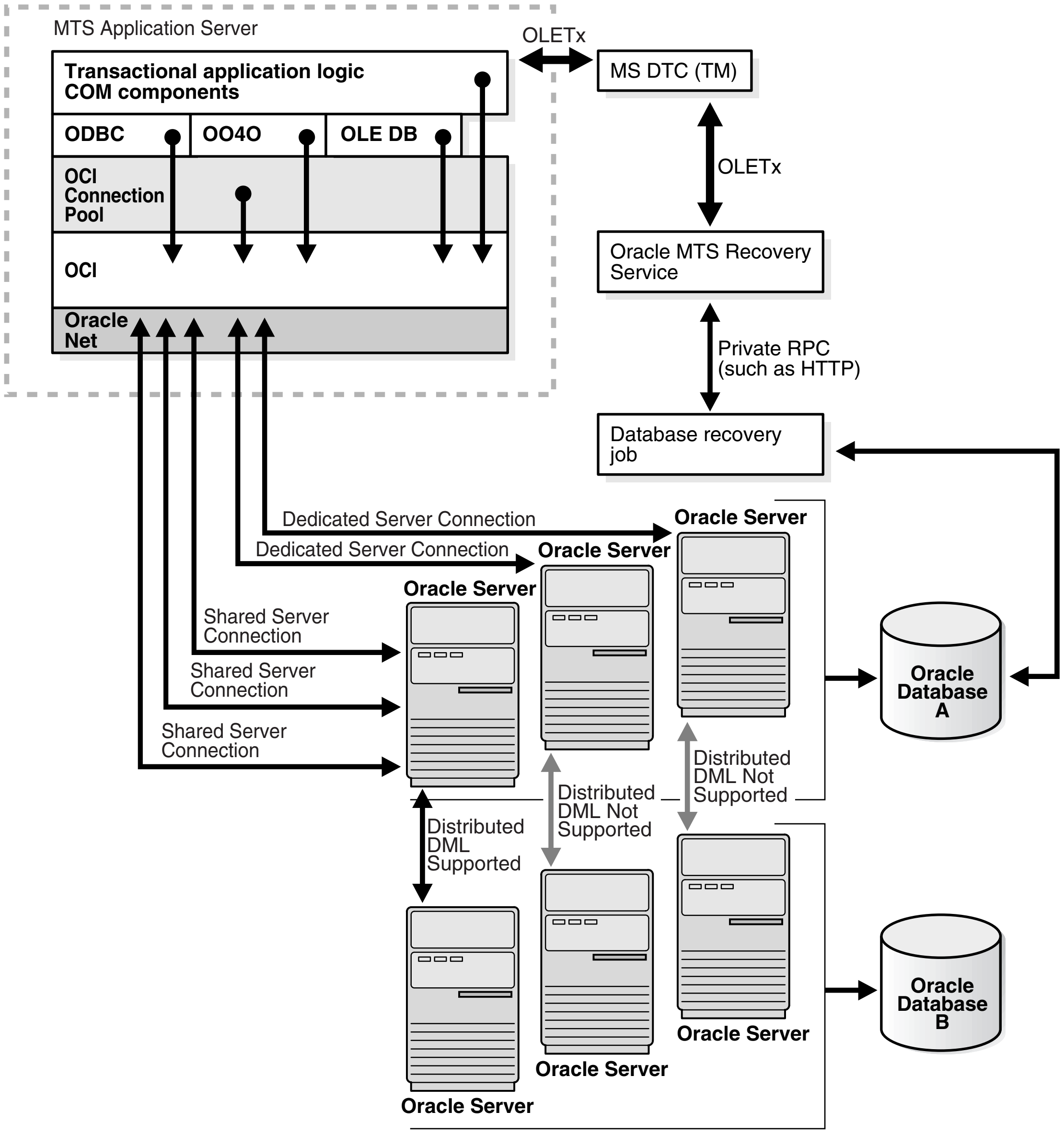
```
SERVER=dedicated
```

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

See Also: *Oracle 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 Oracle database 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 managers (for example, Tuxedo, MS DTC) for coordinating transaction outcome

The two-phase commit protocol completes these transactions. During phase one, the transaction manager (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, 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. If you later change the database with which Microsoft Transaction Server is coordinating transactions, you can drop the administrative user account schema from the previous database.

To drop the Microsoft Transaction Server administrative user account:

1. Start SQL*Plus:

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

2. Connect to the database as `SYSDBA`:

   ```
   SQL> CONNECT / 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`).

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

A
Using Oracle Services for Microsoft Transaction Server on Windows Operating Systems

This appendix describes differences between using **Oracle Services for Microsoft Transaction Server (OraMTS)** on Windows operating systems.

This appendix contains these topics:

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

Note: Oracle Services for Microsoft Transaction Server configuration is the same on Windows 2000, Windows NT, Windows XP and Windows Server 2003.

Microsoft Transaction Server Differences on Windows Platforms

Table A–1 describes the differences between using **Microsoft Transaction Server** on Windows NT Server, Windows 2000, Windows XP and Windows Server 2003.

Table A–1 Microsoft Transaction Server Differences on Windows Platforms

Feature	On Windows NT	On Windows 2000	On Windows XP and Windows Server 2003
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) and as a .NET server.	The functionality of Microsoft Transaction Server is integrated within the operating system as a COM+ server (COM+ transactions) and as a .NET server.
Microsoft Management Console	You must obtain the Microsoft Management Console from Microsoft Corporation.	Microsoft Management Console is automatically included.	Microsoft Management Console is automatically included.
Component object model	COM is the name.	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**	Choose **Start** > **Programs** > **Administrative Tools** > **Component Services**
Transaction Coordinator	**Microsoft Distributed Transaction Coordinator (MS DTC)** is the name.	Microsoft Distributed Transaction Coordinator (MSDTC) is the name.	Microsoft Distributed Transaction Coordinator (MSDTC) is the name.

Table A–1 (Cont.) Microsoft Transaction Server Differences on Windows Platforms

Feature	On Windows NT	On Windows 2000	On Windows XP and Windows Server 2003
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.	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 Component Services window appears.
3. Double-click **Console Root** in the **Component Services** window.
4. Double-click **Component Services**.
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 manager).

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 .NET

Microsoft .NET is a set of Microsoft software technologies used to connect information, people, systems, and devices through web services to each other and to larger applications over the Internet.

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 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 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 Oracle release 1 (9.0.1), this was the username for the Oracle Services for Microsoft Transaction Server.

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 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 Server, as of Oracle release 1 (9.0.1)).

Oracle Call Interface (OCI)

An application programming interface that enables you to manipulate data and schemas in a database. 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 Data Provider for .NET (ODP.NET)

Oracle Data Provider for .NET (ODP.NET) features optimized data access to the Oracle database from a .NET environment. ODP.NET includes support for connection pooling, PL/SQL, LOBs, RefCursors, globalization/localization, proxy user authentication/ parameter array binding, named parameters, and safe type mapping between Oracle types and .NET types.

Oracle Fail Safe

Ensures that if a failure occurs on one cluster node, then the databases 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, as of Oracle release 1 (9.0.1).

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 lets the Oracle MTS Recovery Service resolve in-doubt transactions.

Oracle Objects for OLE (OO4O)

Oracle Objects for OLE (OO4O) is a COM-based database connectivity tool that combines seamless and optimized access to Oracle databases with easy to use interfaces.

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, as of Oracle release 1 (9.0.1).

Oracle Services for Microsoft Transaction Server (OraMTS)

A component that provides full integration of the Oracle database 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 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