# Oracle® TimesTen In-Memory Database

Troubleshooting Guide

Release 18.1

**E61198-06**

July 2020

Oracle TimesTen In-Memory Database Troubleshooting Guide, Release 18.1

E61198-06

# Contents

## 1 Tools for Troubleshooting TimesTen

## 2 Troubleshooting TimesTen Applications and Databases

## 3 Troubleshooting TimesTen Application-Tier Database Cache

## 4 Troubleshooting Replication

## 5 Troubleshooting AWT Cache Groups

## Index

# Preface

Oracle TimesTen In-Memory Database (TimesTen) is a relational database that is memory-optimized for fast response and throughput. The database resides entirely in memory at runtime and is persisted to the file system.

- Oracle TimesTen In-Memory Database in classic mode, or TimesTen Classic, refers to single-instance and replicated databases (as in previous releases).
- Oracle TimesTen In-Memory Database in grid mode, or TimesTen Scaleout, refers to a multiple-instance distributed database. TimesTen Scaleout is a grid of interconnected hosts running instances that work together to provide fast access, fault tolerance, and high availability for in-memory data.
- TimesTen alone refers to both classic and grid modes (such as in references to TimesTen utilities, releases, distributions, installations, actions taken by the database, and functionality within the database).
- TimesTen Application-Tier Database Cache, or TimesTen Cache, is an Oracle Database Enterprise Edition option. TimesTen Cache is ideal for caching performance-critical subsets of an Oracle database into cache tables within a TimesTen database for improved response time in the application tier. Cache tables can be read-only or updatable. Applications read and update the cache tables using standard Structured Query Language (SQL) while data synchronization between the TimesTen database and the Oracle database is performed automatically. TimesTen Cache offers all of the functionality and performance of TimesTen Classic, plus the additional functionality for caching Oracle Database tables.
- TimesTen Replication features, available with TimesTen Classic or TimesTen Cache, enable high availability.

TimesTen supports standard application interfaces JDBC, ODBC, and ODP.NET; Oracle interfaces PL/SQL, OCI, and Pro*C/C++; and the TimesTen TTClasses library for C++.

## Audience

This guide describes how to troubleshoot some of the problems users encounter.

To work with this guide, you should understand how database systems work and have some knowledge of SQL (Structured Query Language).

## Related documents

TimesTen documentation is available at:

`https://docs.oracle.com/database/timesten-18.1`

Oracle Database documentation is also available on the Oracle documentation website. This may be especially useful for Oracle Database features that TimesTen supports but does not attempt to fully document.

## Conventions

TimesTen supports multiple platforms. Unless otherwise indicated, the information in this guide applies to all supported platforms. The term Windows refers to all supported Windows platforms. The term UNIX applies to all supported UNIX platforms. The term Linux is used separately. Refer to "Platforms and compilers" in *Oracle TimesTen In-Memory Database Release Notes* (`README.html`) in your installation directory for specific platform versions supported by TimesTen.

> **Note:** In TimesTen documentation, the terms "data store" and "database" are equivalent. Both terms refer to the TimesTen database.

This document uses the following text conventions:

| Convention | Meaning |
|---|---|
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |
| *`italic monospace`* | Italic monospace type indicates a placeholder or a variable in a code example for which you specify or use a particular value. For example:<br>`LIBS = -Ltimesten_home/install/lib -ltten`<br>Replace *`timesten_home`* with the path to the TimesTen instance home directory. |
| [ ] | Square brackets indicate that an item in a command line is optional. |
| { } | Curly braces indicated that you must choose one of the items separated by a vertical bar ( \| ) in a command line. |
| \| | A vertical bar (or pipe) separates alternative arguments. |
| . . . | An ellipsis (. . .) after an argument indicates that you may use more than one argument on a single command line. |
| % or $ | The percent sign or dollar sign indicates the UNIX shell prompt, depending on the shell that is used. |
| # | The number (or pound) sign indicates the UNIX root prompt. |

TimesTen documentation uses these variables to identify path, file and user names:

| Convention | Meaning |
|---|---|
| *`installation_dir`* | The path that represents the directory where the current release of TimesTen is installed. |
| *`timesten_home`* | The path that represents the home directory of a TimesTen instance. |

| Convention | Meaning |
|---|---|
| *`release`* or *`rr`* | The first two parts in a release number, with or without dots. The first two parts of a release number represent a major TimesTen release. For example, 181 or 18.1 represents TimesTen Release 18.1. |
| *`DSN`* | The data source name. |

**Note:** TimesTen release numbers are reflected in items such as TimesTen utility output, file names and directory names. The release numbers for these items are subject to change with every minor or patch release, and the documentation cannot always be up to date. The documentation seeks primarily to show the basic form of output, file names, directory names and other code that may include release numbers. You can confirm the current release number by looking at the Release Notes or executing the `ttVersion` utility.

# Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

## Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

# What's New

This section summarizes the new features and functionality of TimesTen Release 18.1 that are documented in this guide, providing links into the guide for more information.

## New features in Release 18.1.1.1.0

Oracle TimesTen In-Memory Database in classic mode or TimesTen Classic refers to single-instance environments and databases as in previous releases. The information in this book is for TimesTen Classic.

# 1

# Tools for Troubleshooting TimesTen

The following sections in this chapter describe how to use the TimesTen utilities and other tools that are used to diagnose problems with the TimesTen database:

- Using the ttCapture utility
- Using the ttIsql utility
- Using the ttStatus utility
- Using the logs generated by the TimesTen daemon
- Using the ttTraceMon utility
- Using the ttXactAdmin utility
- Using ODBC tracing
- Monitoring the TimesTen system tables
- Using the query optimizer

## Using the ttCapture utility

The `ttCapture` utility captures information about the configuration and state of TimesTen. The `ttCapture` utility generates a file named `ttcapture.`*`date`*`.`*`time`*`.log`. By default, the file is written to the directory from which you invoke the `ttCapture` utility. Use the `ttCapture -dest` option to direct the output file to be written to another directory.

If you run `ttCapture` again, it writes the information to a new file.

When you experience a problem with a TimesTen database, run `ttCapture` with the *`DSN`* option for the database as soon as possible, either when you are encountering the problem or immediately afterward.

**Note:** Always double-quote directory and file names in case there are spaces in the names.

When you contact TimesTen Customer Support, upload the `ttcapture.`*`date`*`.`*`number`*`.log` file to the Service Request.

See "ttCapture" in the *Oracle TimesTen In-Memory Database Reference* for information about additional options.

## Using the ttIsql utility

The `ttIsql` utility enables you to interactively execute SQL statements and report status information on your TimesTen database.

All TimesTen SQL operations can be executed from a `ttIsql` `Command>` prompt.

***Example 1–1 Using the ttIsql utility***

To start the `ttIsql` utility for the `database1` database, enter:

```
% ttIsql database1
```

You should see output similar to the following:

```
Copyright (c) 1996, 2019, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.

connect "DSN=database1";
Connection successful:
DSN=database1;UID=testuser;DataStore=/scratch/testuser/database1;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;PermSize=128;
(Default setting AutoCommit=1)
```

You can then execute SQL statements or `ttIsql` commands at the `Command>` prompt. You can also call TimesTen built-in procedures. One useful procedure is `ttConfiguration`. When you call `ttConfiguration`, TimesTen displays the values of the connection attributes for your current database connection. For more information, see "ttConfiguration" in the *Oracle TimesTen In-Memory Database Reference*.

"Using the ttIsql Utility" in the *Oracle TimesTen In-Memory Database Operations Guide* describes how to use the most common `ttIsql` commands. The following `ttIsql` commands are commonly used when troubleshooting:

- `dssize` prints database size information.

  See "Displaying database structure information" in the *Oracle TimesTen In-Memory Database Operations Guide*.

- `showplan` prints the optimizer execution plans for selects, updates, and deletes in this transaction.

  See "Viewing and changing query optimizer plans" in the *Oracle TimesTen In-Memory Database Operations Guide*.

- `timing` prints query timing.

  See "Timing ODBC function calls" in the *Oracle TimesTen In-Memory Database Operations Guide*.

- `optprofile` prints the current optimizer flag settings and join order.

  See "Viewing and changing query optimizer plans" in the *Oracle TimesTen In-Memory Database Operations Guide*.

For the full list of `ttIsql` features, see "ttIsql" in the *Oracle TimesTen In-Memory Database Reference*.

## Using the ttStatus utility

Use the `ttStatus` utility to check the status of the TimesTen daemon and the state of all TimesTen connections.

**Example 1–2 ttStatus shows TimesTen daemon is not running**

In this example, the output from `ttStatus` indicates that no TimesTen daemon is running. If the daemon has stopped unexpectedly, see "No response from TimesTen daemon or subdaemon" on page 2-2 for troubleshooting information.

On Linux and UNIX systems:

```
% ttStatus
ttStatus: Could not connect to the TimesTen daemon.
If the TimesTen daemon is not running, please start it
by running "ttDaemonAdmin -start".
```

**Example 1–3 ttStatus shows TimesTen daemon is running**

In this example, the output from `ttStatus` indicates that the TimesTen daemon is running. There is one database named `database1`.

The first line indicates that the TimesTen daemon is running as process 26251 on port 6624 for the `instance1` TimesTen instance. The second line indicates the TimesTen Server is running as process 26258 on port 6625.

There are currently 14 connections to the database: one user and 13 subdaemon connections.

The restart policies for the cache agent and the replication agent in the database are set to `manual`.

```
% ttStatus
TimesTen status report as of Tue Mar 19 10:16:07 2019

Daemon pid 26251 port 6624 instance instance1
TimesTen server pid 26258 started on port 6625
------------------------------------------------------------------------
------------------------------------------------------------------------
Data store /scratch/testuser/database1
Daemon pid 26251 port 6624 instance instance1
TimesTen server pid 26258 started on port 6625
There are 14 connections to the data store
Shared Memory KEY 0x021001b9 ID 10813443
PL/SQL Memory KEY 0x031001b9 ID 10846212 Address 0x5000000000
Type            PID     Context             Connection Name              ConnID
Process         26291   0x00007f5975c07010  database1                         1
Subdaemon       26256   0x00007fe2000008c0  Deadlock Detector              2043
Subdaemon       26256   0x00007fe2040008c0  Checkpoint                     2041
Subdaemon       26256   0x00007fe2080008c0  Monitor                        2045
Subdaemon       26256   0x00007fe2280008c0  Flusher                        2044
Subdaemon       26256   0x00007fe230eb1010  Garbage Collector              2037
Subdaemon       26256   0x00007fe2312ce010  XactId Rollback                2038
Subdaemon       26256   0x00007fe2317ec010  IndexGC                        2035
Subdaemon       26256   0x00007fe231d0a010  HistGC                         2039
Subdaemon       26256   0x00007fe232228010  AsyncMV                        2040
Subdaemon       26256   0x00007fe232746010  Log Marker                     2036
Subdaemon       26256   0x00007fe232c64010  Aging                          2042
Subdaemon       26256   0x00007fe238170010  Rollback                       2046
Subdaemon       26256   0x00007fe23858d010  Manager                        2047
Replication policy  : Manual
Cache Agent policy  : Manual
PL/SQL enabled.
------------------------------------------------------------------------
Accessible by group g900
End of report
```

***Example 1–4 ttStatus shows connection to old instance***

This example shows a connection to an old instance of a database. This can occur when a database is invalidated, but some users have not disconnected from the invalidated copy of the database still in memory. After all users disconnect, the memory is automatically freed.

```
% ttStatus

TimesTen status report as of Mon Aug  6 22:03:04 2012
Daemon pid 5088 port 17000 instance instance1
TimesTen server pid 4344 started on port 17002
------------------------------------------------------------------
Data store c:\temp\sample
There are no connections to the data store
Obsolete or not yet active connection(s):
Process  4696 context 0xd08800 name sample connid 1, obsolete connection, shmKey
'Global\DBI45b94c6f.3.SHM.4'
Replication policy : Manual
Cache agent policy : Manual
------------------------------------------------------------------
End of report
```

## Using the logs generated by the TimesTen daemon

The TimesTen daemon and its subdaemons and agents generate error, warning, and informational messages. These messages may be useful for TimesTen system administration and for debugging applications.

By default, informational messages are stored in:

- A user error log that contains information you may need to be aware of, including actions you may need to take.
- A support log containing everything in the user error log plus information for use by TimesTen Customer Support.

See "Error, warning, and informational messages" in the *Oracle TimesTen In-Memory Database Operations Guide* for information about configuring the logs, including their location and size.

All message categories are enabled by default, but you can also disable or explicitly enable the following categories of user error log messages for all databases or specified databases through options of the `ttDaemonLog` utility:

- `ALL` (default): For all messages
- `CACHE`: For messages from the cache agent, designated by `CAC`
- `DAEMON`: For messages from the main daemon and subdaemons
- `DAEMONDBG`: For additional information from the main daemon and subdaemons.
- `REPLICATION`: For messages from the replication agent, designated by `REP`
- `SERVER`: For messages from TimesTen Server.

See "ttDaemonLog" in *Oracle TimesTen In-Memory Database Reference* for information about `ttDaemonLog` options.

## Using the ttTraceMon utility

Use the `ttTraceMon` utility to log various trace information on a number of TimesTen components. Each TimesTen component can be traced at different levels of detail. You can list all of the traceable TimesTen components and their current tracing level by specifying `ttTraceMon` with the `show` subcommand. See "ttTraceMon" in the *Oracle TimesTen In-Memory Database Reference* for a list of options.

TimesTen tracing severely impacts application performance and consumes a great deal of file system space if trace output is directed to a file. In addition, when using AWT cache groups, you must restart the replication agent when trying to trace the `ORACON` component with `ttTraceMon`. Use the `ttTraceMon` utility only when diagnosing problems. When you are finished, reset tracing to the default values.

***Example 1–5 Default trace levels for components***

This example shows that the levels for most tracing components are set to level 0 (off) for the `database1` database. Both the `ERR` and `DEADLOCK` components are set to 1 for tracing by default. See "ERR tracing" on page 1-12.

```
% ttTraceMon -e show database1
AGING         ... 0
API           ... 0
ASYNCMV       ... 0
AUTOREFRESH   ... 0
BULKLOAD      ... 0
BLK           ... 0
CG            ... 0
CKPT          ... 0
DBG0          ... 0
DBG1          ... 0
DEADLOCK      ... 1
DIST          ... 0
EE            ... 0
ERR           ... 1
EXTTBLLOAD    ... 0
FLOW          ... 0
GRID          ... 0
GRIDAS        ... 0
HEAP          ... 0
INTERRUPT     ... 0
IX            ... 0
IXGC          ... 0
IXHEAP        ... 0
LATCH         ... 0
LBCU          ... 0
LOB           ... 0
LOCK          ... 0
LOG           ... 0
LOGF          ... 0
MEM           ... 0
ODBC          ... 0
OPT           ... 0
ORACON        ... 0
PLOAD         ... 0
PREP          ... 0
PT            ... 0
REPL          ... 0
SM            ... 0
SQL           ... 0
```

```
TEST         ... 0
TRACE        ... 1
XA           ... 0
XACT         ... 0
XLA          ... 0
```

The output for most TimesTen components is only of interest to TimesTen Customer Support. However, the output for the `SQL`, `API`, `LOCK`, `ERR`, `AGING` and `AUTOREFRESH` components may be useful to you when you are troubleshooting application problems.

The rest of this section includes the following topics:

- Starting a trace and reading the trace buffer
- SQL tracing
- API tracing
- DEADLOCK tracing
- LOCK tracing
- ERR tracing
- AGING tracing
- AUTOREFRESH tracing

## Starting a trace and reading the trace buffer

Start a new trace by specifying `ttTraceMon` *`datastore`*. For example, to start a trace on the `database1` database, enter:

```
%  ttTraceMon database1
Trace monitor; empty line to exit
Trace >
```

At the `Trace` prompt, specify the type of trace and its level. For example, to start tracing the `SQL` component at level 3, enter:

```
Trace > level sql 3
```

At this point you can run your application and the TimesTen trace information is written to a trace buffer. There are two ways to read the contents of the trace buffer:

- From the `Trace` prompt, use the `outfile` command to direct the trace buffer data to a file. You must do this before running your application. When writing tracing information to a file, new trace information is concatenated to the existing contents of the file.
- From the `Trace` prompt, use the `dump` command to display the trace buffer data to your screen.

**Note:** The contents of the trace buffer accumulate with each new trace. To clear the trace buffer, use the `flush` command from a `ttTraceMon` prompt. Clear the buffered trace records for a specific component by specifying the component with the `flush` command.

Each record from the trace buffer has the following format:

```
timestamp   sequence   component   level   connection   processid
   operation
```

The fields in the records are defined as follows:

- *`timestamp`* is the time at which the operation was executed.
- *`sequence`* is the incremental number that identifies the trace line.
- *`component`* is the TimesTen component being traced (such as `SQL`, `API`, `LOCK`, or `ERR`).
- *`level`* is the trace level associated with the trace line. The range of trace levels differs by component, but for all components, the lowest trace level generates the least verbose output and highest trace level generates the most verbose output. For example, if you are tracing `SQL` at level 4, your output includes lines for levels 2 (prepare), 3 (execute), and 4 (open, close, fetch).

> **Note:** Trace levels for some components are not a continuous range of numbers. If you enter a number that does not correspond to a supported level for a component, tracing occurs at the highest supported level that is less than the number you entered. For example, if tracing levels for a component are 1, 2, 3, 4, and 6 and you enter 5, tracing events for level 1, 2, 3, and 4 are generated.

- *`connection`* is the internal connection ID identifying the connection that generated the trace. This number corresponds to the `ConnID` shown in the `ttStatus` output. The connection ID is also used as the first element of the transaction ID.
- *`processid`* is the operating system process ID for the process that generated the trace.
- *`operation`* is the operation that occurred (such as SQL statement, API operation, or error message).

For example, a line from the trace buffer after a SQL trace at level 3 might look like this:

```
10:39:50.231 5281 SQL 2L 1C 3914P Preparing: select cust_num from customer
```

## SQL tracing

Using `ttTraceMon` with the `SQL` component provides information about the SQL being prepared or executed by the TimesTen engine. Table 1–1 describes the levels for SQL tracing. Each level with a '+' sign includes the trace information described for that level, plus all levels preceding it.

***Table 1–1 SQL tracing levels***

| Level | Output |
|---|---|
| 2 | SQL commands being prepared |
| 3 | + SQL commands being executed |
| 4 | + The effect of command pooling (prepares not being done because the prepared command already exists in the pool), the need for reprepares (for example, because an index was created), and commands being destroyed<br>At this level, `ttTraceMon` also shows when a query command is being opened, fetched, and closed. |
| 5 | + Some internal data, such as command numbers, which are not generally useful for customer-level debugging |

---

**Note:** TimesTen recommends tracing SQL at level 3 or 4. SQL tracing does not show any information about the optimizer. Optimizer tracing is managed by a separate component (`OPT`) at level 4 only, and is not designed for customer use.

---

***Example 1–6 SQL trace***

In this example, execute `ttTraceMon` to do a SQL trace at level 4 on the `database1` database. Direct the output from the SQL trace to the `SQLtrace.txt` file. Then flush the buffer so that the trace does not report past SQL statements.

```
% ttTraceMon database1
Trace monitor; empty line to exit
Trace > outfile SQLtrace.txt
Trace > level sql 4
Trace > flush
```

At this point, execute an application that includes the following SQL statement:

```
SELECT * FROM departments WHERE department_id = 10;
```

The trace information is written to the `SQLtrace.txt` file:

```
12:19:36.582     269 SQL      2L    3C  29570P Preparing: select * from
departments where department_id = 10
12:19:36.583     270 SQL      4L    3C  29570P sbSqlCmdCompile ()(E): (Found
already compiled version: refCount:01, bucket:28) cmdType:100, cmdNum:1000146.
12:19:36.583     271 SQL      4L    3C  29570P Opening: select * from departments
where department_id = 10;
12:19:36.583     272 SQL      4L    3C  29570P Fetching: select * from
departments where department_id = 10;
12:19:36.583     273 SQL      4L    3C  29570P Closing: select * from departments
where department_id = 10;
5 records dumped
```

When the application has completed, turn off SQL tracing and exit `ttTraceMon`.

```
Trace > level sql 0
Trace > {press ENTER - blank line}
```

## API tracing

API traces are generated for database operations such as connecting to a database, changing a connection attribute, and committing a transaction. Table 1–2 describes the levels for API tracing. Each level with a '+' sign includes the trace information described for that level, plus all levels preceding it.

***Table 1–2 API tracing levels***

| Level | Output |
|---|---|
| 1 | All rollback attempts by the subdaemon<br>This occurs if an application exits abruptly and the subdaemon recovers its connection. |
| 2 | + Some low-on-space conditions |
| 3 | + Create, connect, disconnect, checkpoint, backup, and compact operations for the database, as well as commit and rollback for each connection, and a few other operations |

*Table 1–2 (Cont.) API tracing levels*

| Level | Output |
|---|---|
| 4 | + Most other operations conducted at TimesTen's internal API level<br>It does not show numerous operations on the storage manager and indexes that are done internally. |

**Note:** TimesTen recommends tracing at level 3.

*Example 1–7 API trace*

In this example, execute `ttTraceMon` to do a API trace at level 3 on the `databae1` database. The output from the API trace is written to the `APItrace.txt` file. Before saving the API trace to the buffer, use the `flush` command to empty the buffer.

```
% ttTraceMon database1
Trace monitor; empty line to exit
Trace> outfile APItrace.txt
Trace> level api 3
Trace > flush
```

At this point, execute the application. When the application has completed, turn off API tracing and exit `ttTraceMon`:

```
Trace > level api 0
Trace > {press ENTER - blank line}
```

The contents of `APItrace.txt` are similar to the sample output shown below. The output shows connection to the database, setting the connection character set, setting the isolation level, and committing a transaction.

```
11:54:26.796    1016 API      3L    2C   4848P sb_dbConnect()(X)
11:54:26.796    1017 API      3L    2C   4848P sb_dbConnCharsetSet()(E)
11:54:26.796    1018 API      3L    2C   4848P sb_dbConnSetIsoLevel()(E)
11:54:39.795    1019 API      3L    2C   4848P sb_dbConnSetIsoLevel()(E)
11:54:45.253    1020 API      3L    2C   4848P sb_xactCommitQ()(E)
```

## DEADLOCK tracing

Use the `DEADLOCK` component to trace the occurrences of deadlocks for all applications.

Table 1–3 describes the `DEADLOCK` tracing levels. Each level with a '+' sign includes the trace information described for that level, plus all levels preceding it.

*Table 1–3 DEADLOCK tracing levels*

| Level | Output |
|---|---|
| 1 | Deadlock cycles, logged as they are discovered |
| 4, 6 | + Detailed information about how the deadlock is detected |

*Example 1–8 DEADLOCK trace*

In this example, execute `ttTraceMon` to do a `DEADLOCK` trace at level 1, which is the default, on `myDSN` database.

Make two connections to `myDSN`. For the first connection, autocommit is on. Create table `test` and insert two rows. Then, set autocommit off and update the `x1=1` row of table `test`. Because autocommit is off, the row is not inserted into the table until the commit. A lock is held until the transaction is committed or rolled back.

```
Command> create table test (x1 int unique, y1 int);
Command> insert into test values (1,1);
1 row inserted.
Command> insert into test values (2,2);
1 row inserted.
Command> autocommit 0;
Command> update test set y1=y1 where x1=1;
1 row updated.
```

For the second connection to `myDSN`, autocommit is set to off. Update the `x1=2` row of table `test`.

```
Command> autocommit 0;
Command> update test set y1=y1 where x1=2;
1 row updated.
```

Now, create a deadlock situation by executing update statements in both connections for rows that are locked by each other. The first connection executes an update against the row where `x1=2`.

```
Command> update test set y1=y1 where x1=2;
 6003: Lock request denied because of time-out
Details: Tran 2.1 (pid 32750) wants Un lock on rowid BMUFVUAAAAaAAAAETk,
table ME.TEST. But tran 3.2 (pid 32731) has it in Xn (request was Xn).
Holder SQL (update t1 set y1=y1 where x1=2)
The command failed.
```

The second connection executes an update against the row where `x1=1`.

```
Command> update test set y1=y1 where x1=1;
 6002: Lock request denied because of deadlock
Details: Tran 3.2 (pid 32731) wants Un lock on rowid BMUFVUAAAAaAAAADzk,
table ME.TEST. But tran 2.1 (pid 32750) has it in Xn (request was Xn).
Holder SQL (update t1 set y1=y1 where x1=1)
The command failed.
```

Use the `flush` command to empty the buffer.

```
% ttTraceMon myDSN
Trace monitor; empty line to exit
Trace> flush
```

The trace buffer contains the following information showing all level 1 deadlock traces, as evidenced by '`1L`'.

```
Trace> dump
09:50:26.444      13 DEADLOCK 1L 2036C   3484P edge 1: xid 3.2, cid 3,
<Row BMUFVUAAAAaAAAADzk,0x8c5
74(574836)> 0 cnt=1 , Tbl 'T1', SQL='update t1 set y1=y1 where x1=1'
09:50:26.455      14 DEADLOCK 1L 2036C   3484P edge 0: xid 2.1, cid 2,
<Row BMUFVUAAAAaAAAAETk,0x8c5
74(574836)> 0 cnt=1 , Tbl 'T1', SQL='update t1 set y1=y1 where x1=2'
09:50:26.455      15 DEADLOCK 1L 2036C   3484P Victim: xcb:3.2,
SQL='update t1 set y1=y1 where x1=1'
```

If you want more information, set `DEADLOCK` tracing to a higher value. For example, the following sets `DEADLOCK` tracing to level 4 in `ttTraceMon`:

```
Trace > level deadlock 4
```

## LOCK tracing

Use the `LOCK` component to trace the locking behavior of your application to detect trouble with deadlocks or lock waits. `LOCK` tracing generates a great deal of volume, so it is important to choose the appropriate level of tracing. Level 3, for example, begins to generate a large number of traces, as traces are written for fairly common events. Also, the traces themselves may be hard to read in places. If you cannot discern enough information, contact TimesTen Customer Support for more information.

Table 1–4 describes the `LOCK` tracing levels. Each level with a '+' sign includes the trace information described for that level, plus all levels preceding it.

***Table 1–4 LOCK tracing levels***

| Level | Output |
|---|---|
| 1 | Deadlock cycles as they are discovered |
| 2 | + Failures to grant locks for any reason |
| 3 | + Lock waits (which may or may not be granted) |
| 4 | + All lock grants/releases, some routine calls, and the mechanism of the deadlock detector |
| 6 | + Each step in cycle traversal |

***Example 1–9 LOCK trace***

In this example, execute `ttTraceMon` to do a `LOCK` trace at level 4 on `myDSN` database.

Make two connections to `myDSN`. For the first connection, set autocommit on. Create table `test` and insert three rows. Create a materialized view using table `test`.

Turn on tracing at level 4 and use the `flush` command to empty the buffer.

```
% ttTraceMon myDSN
Trace monitor; empty line to exit
Trace> level lock 4
Trace> flush
```

For the second connection to `myDSN`, set autocommit off. Insert a row into table `test`. Because autocommit is off, the row is not inserted into the table until the commit. A lock is held until the transaction is committed or rolled back.

If the `dump` command is used to display the contents of the trace buffer, there are no records in the trace buffer:

```
Trace> dump
0 records dumped
```

From the first connection, try to drop the materialized view. The view cannot be dropped because there is a transaction that has not been committed or rolled back:

```
Command> drop materialized view v;
 6003: Lock request denied because of time-out
Details: Tran 3.71 (pid 24524) wants Sn lock on table TTUSER.TEST. But tran 1.42
(pid 24263) has it in IXn (request was IXn). Holder SQL (insert into test
values (100);)
The command failed.
```

The trace buffer contains the following information:

```
Trace> dump
20:09:04.789  174759 LOCK     3L    3C  24524P ENQ: xcb:00003 <Tbl 0x9b894,0x0>
```

```
0+Sn=>SL activity 0 Sn cnt=0; Holder xcb:00001 IXn
20:09:04.789  174760 LOCK     3L    3C  24524P Connection 3 scheduled for sleep
20:09:04.789  174761 LOCK     3L    3C  24524P Connection 3 sleeping
20:09:14.871  174762 LOCK     3L 2047C  24237P Connection 3 timed out
20:09:14.871  174763 LOCK     3L 2047C  24237P Connection 3 woken up
20:09:14.871  174764 LOCK     3L    3C  24524P Connection 3 awake
20:09:14.871  174765 LOCK     2L    3C  24524P ENQ: xcb:00003 <Tbl 0x9b894,0x0>
0+Sn=>TM activity 0 Sn cnt=1; Holder xcb:00001 IXn
7 records dumped
```

When finished with the trace, set `LOCK` tracing back to its default setting (0) and exit `ttTraceMon`:

```
Trace > level lock 0
Trace > {press ENTER - blank line}
```

## ERR tracing

It may be useful to trace the `ERR` component. For example, an `ERR` trace at level 4 shows all of the error messages that are pushed in the TimesTen direct driver (not all errors are output to the user because they are handled internally). `ERR` tracing at level 1 is the default. No output is written for `ERR` tracing at level 2 and 3.

Table 1–5 describes `ERR` tracing levels. Each level with a '+' sign includes the trace information described for that level, plus all levels preceding it.

***Table 1–5 ERR tracing levels***

| Level | Output |
|---|---|
| 1 (set by default) | Fatal errors |
| 4 | + All other error messages, many of which are handled internally by TimesTen |

***Example 1–10 ERR trace***

In this example, execute `ttTraceMon` to do a `ERR` trace at level 4 on `myDSN` database.

First, create a table:

```
Command> CREATE TABLE test (id TT_INTEGER);
```

Next, turn on tracing at level 4. Rather than direct the trace output to a file as in the previous examples, read it directly from the trace buffer. Before saving the `ERR` trace to the buffer, use the `flush` command to empty the buffer.

```
% ttTraceMon myDSN
Trace monitor; empty line to exit
Trace> level err 4
Trace> flush
```

Now execute a SQL script with three errors in it. The errors are:

- Creating a table with the same name as an existing table
- Using incorrect syntax to insert values into the table
- Inserting `CHAR` data into a `TT_INTEGER` column

```
Command> CREATE TABLE test (id TT_INTEGER);
 2207: Table TEST already exists
The command failed.
Command> INSERT INTO test VALUES 'abcd');
```

```
 1001: Syntax error in SQL statement before or at: "'abcd'", character position:
 25
insert into test values 'abcd');
                        ^^^^^^
The command failed.
Command> INSERT INTO test VALUES ('abcd');
 2609: Incompatible types found in expression
The command failed.
```

The trace information is written to the trace buffer. Display it by using the `dump` command.

```
Trace> dump
19:28:40.465  174227 ERR      4L    1C  24263P TT2207: Table TEST already exists
 -- file "eeDDL.c", lineno 2930, procedure "sbEeCrDtblEval()"
19:28:51.399  174228 ERR      4L    1C  24263P TT1001: Syntax error in SQL
statement before or at: "'abcd'", character position: 25
insert into test values 'abcd');
                        ^^^^^^
 -- file "ptSqlY.y", lineno 6273, procedure "reserved_word_or_syntax_error"
19:29:00.725  174229 ERR      4L    1C  24263P TT2609: Incompatible types found
in expression -- file "saCanon.c", lineno 12618, procedure "sbPtAdjustType()"
3 records dumped
```

Set ERR tracing back to its default setting (1) and exit `ttTraceMon`:

```
Trace > level err 1
Trace > {press ENTER - blank line}
```

## AGING tracing

Use the `ttTraceMon` utility to obtain the following information:

- When aging starts and ends
- How many rows have been deleted by the aging subdaemon

See "Implementing aging in your tables" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Table 1–6 describes the `AGING` tracing levels. Each level with a '+' sign includes the trace information described for that level, plus all levels preceding it.

***Table 1–6 AGING tracing levels***

| Level | Description |
|---|---|
| 1 | Messages about the following events:<br>- The aging subdaemon starts least recently used (LRU) or time-based aging.<br>- The aging subdaemon repeats LRU aging because the LRU threshold was not met.<br>- The aging subdaemon ends LRU or time-based aging. |
| 2 | + Messages about the following events *for each table*:<br>- Aging has started.<br>- Aging has ended. The message includes the reason for ending and the total number of rows deleted. |
| 3 | + Detailed report on how many rows were deleted during each aging cycle |
| 4 | + Message every time the aging subdaemon wakes up |

***Example 1–11 AGING trace***

In this example, execute `ttTraceMon` to do an `AGING` trace on `myDSN` database. The database contains `TTUSER.MYTAB` table, which has a time-based aging policy. The table is described as follows:

```
Command> DESCRIBE ttuser.mytab;

Table TTUSER.MYTAB:
  Columns:
   *ID                              TT_INTEGER NOT NULL
    TS                              TIMESTAMP (6) NOT NULL
  Aging use TS lifetime 3 minutes cycle 1 minute on

1 table found.
(primary key columns are indicated with *)
```

The table contains the following rows before the aging cycle begins:

```
Command> select * from TTUSER.MYTAB;
< 1, 2007-03-21 12:54:06.000000 >
< 3, 2010-03-17 08:00:00.000000 >
< 4, 2007-03-21 12:59:40.000000 >
< 5, 2007-03-21 13:00:10.000000 >
< 6, 2007-03-21 13:01:22.000000 >
5 rows found.
```

Execute `ttTraceMon` to do an `AGING` trace at level 3. Rather than direct the trace output to a file, read it directly from the trace buffer. Before saving the `AGING` trace to the buffer, use the `flush` command to empty the buffer.

```
% ttTraceMon myDSN
Trace monitor; empty line to exit
Trace> level aging 3
Trace> flush
```

Display the trace information in the buffer by using the `dump` command.

```
Trace> dump
13:16:56.802    1247 AGING    1L 2045C  17373P Entering sbAgingTB(): curTime=78
13:16:56.803    1248 AGING    2L 2045C  17373P Entering sbAgingOneTable():
curTime=78, ltblid= 637140
13:16:56.804    1249 AGING    3L 2045C  17373P curTime=78, 4 deleted, 1
remaining, tbl = TTUSER.MYTAB
13:16:56.804    1250 AGING    2L 2045C  17373P Exiting sbAgingOneTable():
curTime=78, reason = 'no more rows', 4 deleted, 1 remaining, tbl = TTUSER.MYTAB
13:16:56.804    1251 AGING    1L 2045C  17373P Exiting sbAgingTB(): curTime=78
5 records dumped
```

Set `AGING` tracing back to its default setting (0) and exit `ttTraceMon`:

```
Trace > level aging 0
Trace > {press ENTER - blank line}
```

## AUTOREFRESH tracing

Use the `ttTraceMon` utility to obtain information about the progress of autorefresh operations for cache groups with the `AUTOREFRESH` cache group attribute.

See "AUTOREFRESH cache group attribute" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Table 1–7 describes `AUTOREFRESH` tracing levels. Each level with a '+' sign includes the trace information described for that level, plus all levels preceding it.

*Table 1–7 AUTOREFRESH tracing levels*

| Level | Description |
|---|---|
| 1 | Autorefresh summary for the interval:<br>■ The time that autorefresh started<br>■ Number of autorefreshed rows for the interval<br>■ Number of autorefreshed root table rows for interval<br>■ Total number of autorefreshed rows since the cache agent started<br>■ Total number of autorefreshed rows in the root table since the cache agent started<br>■ The time that autorefresh ended<br>**Note**: Times and information about root table rows are reported for full autorefresh. |
| 2 | + Autorefresh summary at the cache group level:<br>■ The time that autorefresh started for each cache group<br>■ Number of autorefreshed rows for each cache group<br>■ Number of autorefreshed root table rows for each cache group<br>■ Total number of autorefreshed rows for each cache group since the cache agent started<br>■ Total number of autorefreshed rows in the root table for each cache group since the cache agent started<br>■ The time that autorefresh ended for each cache group<br>**Note**: Times and information about root table rows are reported for full autorefresh. |
| 3 | + Autorefresh summary at the table level:<br>■ The time that autorefresh started<br>■ Number of autorefreshed rows<br>■ Total number of autorefreshed rows since the cache agent started<br>■ The time that autorefresh ended |
| 4 | + Autorefresh details for each table:<br>■ The time that autorefresh started for each table<br>■ The autorefresh query<br>■ Query execute time in milliseconds on the Oracle database<br>■ Query fetch time in milliseconds on the Oracle database<br>■ Query apply time in milliseconds on TimesTen<br>■ Query execute time in milliseconds on the Oracle database for child tables<br>■ Query fetch time in milliseconds on the Oracle database for child tables<br>■ Query apply time in milliseconds on TimesTen for child tables<br>■ The time that autorefresh ended for each table<br>■ The autorefresh bookmark (`logseq`) to which autorefresh was completed |

### *Example 1–12 AUTOREFRESH trace*

In this example, use the `ttTraceMon` utility to trace autorefresh operations on the `cgDSN` database. When the trace level is set to 1, the displayed information is similar to the output of the `ttCacheAutorefreshStatsGet` built-in procedure.

```
% tttracemon cgDSN
Trace monitor; empty line to exit
Trace> level autorefresh 1
Trace> dump

08:56:57.445 19398 AUTOREFRESH 1L 5C 32246P Autorefresh number 1415 started
for interval 60000
08:56:57.883 19419 AUTOREFRESH 1L 5C 32246P Duration For Interval 60000ms: 420
08:56:57.883 19420 AUTOREFRESH 1L 5C 32246P Num Rows For Interval 60000ms: 0
08:56:57.883 19421 AUTOREFRESH 1L 5C 32246P Num Root Rows For Interval
60000ms: 0
08:56:57.883 19422 AUTOREFRESH 1L 5C 32246P Cumulative Rows for Interval
60000ms: 11587
08:56:57.883 19423 AUTOREFRESH 1L 5C 32246P Cumulative Root Rows for Interval
60000ms: 1697
08:56:57.883 19424 AUTOREFRESH 1L 5C 32246P Autorefresh number 1415 ended for
interval 60000ms successfully.
7 records dumped
```

Tracing at level 4 produces additional information about autorefresh operation 1415. Information about autorefresh is provided for the `testuser.readcache` cache group, the `testuser.readtab` root table and the autorefresh interval.

```
Trace> level autorefresh 4
Trace> dump

08:56:57.445 19398 AUTOREFRESH 1L 5C 32246P Autorefresh number 1415 started for
interval 60000
08:56:57.471 19399 AUTOREFRESH 2L 5C 32246P Autorefresh started for cachegroup
TESTUSER.READCACHE
08:56:57.471 19400 AUTOREFRESH 3L 5C 32246P Incremental autorefresh started for
table TESTUSER.READTAB
08:56:57.471 19401 AUTOREFRESH 4L 5C 32246P Autorefresh Query: SELECT L."COL_10",
X."COL_20", X.ft$NotDelete, Z.rowid FROM (SELECT DISTINCT "COL_10" FROM
"TESTUSER"."TT_06_454854_L" WHERE logseq >:logseq AND ft_cacheGroup <>
100000000000*1844259679+-299200618) L,(SELECT "TESTUSER"."READTAB"."COL_10",
"TESTUSER"."READTAB"."COL_20", 1 AS ft$NotDelete  FROM "TESTUSER"."READTAB",
"TESTUSER"."T1" WHERE "TESTUSER"."READTAB"."COL_10" = "TESTUSER"."T1"."COL_10")
X, "TESTUSER"."READTAB" Z WHERE L ."COL_10" = X."COL_10" (+) AND X."COL_10" =
Z."COL_10" (+), logseq: 7
08:56:57.870 19402 AUTOREFRESH 3L 5C 32246P Duration for table
TESTUSER.READTAB: 70
08:56:57.870 19403 AUTOREFRESH 3L 5C 32246P Num Rows for table
TESTUSER.READTAB: 1
08:56:57.870 19404 AUTOREFRESH 3L 5C 32246P Cumulative rows for table
TESTUSER.READTAB: 1559
08:56:57.870 19405 AUTOREFRESH 4L 5C 32246P Autorefresh Query Execute duration
for table TESTUSER.READTAB: 60
08:56:57.870 19406 AUTOREFRESH 4L 5C 32246P Autorefresh Query Fetch duration for
table TESTUSER.READTAB: 0
08:56:57.870 19407 AUTOREFRESH 4L 5C 32246P Autorefresh Query Apply duration for
table TESTUSER.READTAB: 0
08:56:57.870 19408 AUTOREFRESH 4L 5C 32246P Max logseq applied for table
TESTUSER.READTAB: 8
08:56:57.870 19409 AUTOREFRESH 4L 5C 32246P Autorefresh Query Execute duration
```

```
for 7 child(ren) table(s): 32
08:56:57.870 19410 AUTOREFRESH 4L 5C 32246P Autorefresh Query Fetch duration for
7 child(ren) table(s): 0
08:56:57.870 19411 AUTOREFRESH 4L 5C 32246P Autorefresh Query Apply duration for
7 child(ren) table(s): 0
08:56:57.870 19412 AUTOREFRESH 3L 5C  32246P Incremental autorefresh ended for
table TESTUSER.READTAB
08:56:57.872 19413 AUTOREFRESH 2L 5C 32246P Duration For Cache Group
TESTUSER.READCACHE: 1020
08:56:57.872 19414 AUTOREFRESH 2L 5C 32246P Num Rows For Cache Group
TESTUSER.READCACHE: 1
08:56:57.872 19415 AUTOREFRESH 2L 5C 32246P Num Root Rows For Cache Group
TESTUSER.READCACHE: 0
08:56:57.872 19416 AUTOREFRESH 2L 5C 32246P Cumulative Rows for Cache Group
TESTUSER.READCACHE: 11776
08:56:57.872 19417 AUTOREFRESH 2L 5C 32246P Cumulative Root Rows for Cache Group
TESTUSER.READCACHE: 1697
08:56:57.872 19418 AUTOREFRESH 2L 5C 32246P Autorefresh ended for cache group
TESTUSER.READCACHE
08:56:57.883 19419 AUTOREFRESH 1L 5C 32246P Duration For Interval 60000ms: 420
08:56:57.883 19420 AUTOREFRESH 1L 5C 32246P Num Rows For Interval 60000ms: 0
08:56:57.883 19421 AUTOREFRESH 1L 5C 32246P Num Root Rows For Interval
60000ms: 0
08:56:57.883 19422 AUTOREFRESH 1L 5C 32246P Cumulative Rows for Interval
60000ms: 11587
08:56:57.883 19423 AUTOREFRESH 1L 5C 32246P Cumulative Root Rows for Interval
60000ms: 1697
08:56:57.883 19424 AUTOREFRESH 1L 5C 32246P Autorefresh number 1415 ended for
interval 60000ms successfully.
27 records dumped
```

Set `AUTOREFRESH` tracing back to its default setting (0) and exit `ttTraceMon`:

```
Trace > level autorefresh 0
Trace > {press ENTER - blank line}
```

## Using the ttXactAdmin utility

The `ttXactAdmin` utility displays ownership, status, log and lock information for each outstanding transaction. You can also use it to show all current connections to a database. The `ttXactAdmin` utility is useful for troubleshooting problems with replication, XLA, and asynchronous writethrough cache groups.

***Example 1–13 Using ttXactAdmin to diagnose a lock timeout***

Use `ttXactAdmin` to diagnose a lock timeout. Consider two connections that are trying to update the same row. The following transaction by Connection 1 is in progress:

```
UPDATE table1 SET c1 = 2 WHERE c1 = 1;
```

Connection 2 attempts to make the following update:

```
UPDATE table1 SET c1 = 3 WHERE c1 = 1;
```

Connection 2 receives the following error:

```
6003: Lock request denied because of time-out
  Details: Tran 2.3 (pid 2880) wants Un lock on rowid 0x00156bbc, table
 TTUSER.TABLE1.
  But tran 1.21 (pid 2564) has it in Xn (request was Xn). Holder SQL
(update table1 set c1 = 2 where c1 = 1;)
```

```
  The command failed.
```

The details of the error indicate that transaction 1.21 has a lock on row 0x00156bbc, the row that transaction 2.3 wants to update. `ttXactAdmin` displays this information in output that pertains to actions in the entire database:

```
% ttXactAdmin myDSN
2011-03-07 12:57:41.237
c:\datastore\myDSN
TimesTen Release 11.2.2.0.0

Outstanding locks

PID   Context   TransID  TransStatus  Resource   ResourceID   Mode Name

Program File Name: ttIsql

2564  0xeeb9a8  1.21     Active       Database   0x01312d00   IX
                                      Row        0x00156bbc   Xn   TTUSER.TABLE1
                                      Table      1910868      IXn  TTUSER.TABLE1

Program File Name: ttIsql

2880  0xeeb9a8  2.3      Active        Database   0x01312d00  IX
                                       Table      1910868     IXn  TTUSER.TABLE1
                                       Command    19972120    S

Awaiting locks

PID  Context   TransID Resource ResourceID  RMode HolderTransID HMode Name
2880 0xeeb9a8  2.3     Row      0x00156bbc  Un    1.21          Xn TTUSER.TABLE1

2 outstanding transactions found
```

See "ttXactAdmin" in the *Oracle TimesTen In-Memory Database Reference*.

## Using ODBC tracing

On Linux and UNIX systems, ODBC tracing is available only when using a driver manager. To turn on tracing, set the `Trace` and `TraceFile` attributes.

## Monitoring the TimesTen system tables

Each TimesTen database contains a group of system tables that store metadata about the current state of the database. The system tables are described in "System Tables and Views" in the *Oracle TimesTen In-Memory Database System Tables and Views Reference*.

**Note:** You can execute `SELECT` statements on a system table, but you cannot execute a statement such as `INSERT`, `UPDATE` or `DELETE` on these tables.

Use the system table `SYS.SYSTEMSTATS` to view database metrics. In addition, the `ttStats` utility enables you to monitor database metrics and to take and compare snapshots of metrics. For more information, see "SYS.SYSTEMSTATS" in the *Oracle TimesTen In-Memory Database System Tables and Views Reference*, "ttStats" in the *Oracle*

*TimesTen In-Memory Database Reference*, and "TT_STATS" in the *Oracle TimesTen In-Memory Database PL/SQL Packages Reference*.

You can also view the `SYS.MONITOR` table, which contains statistics about certain events that have occurred since the first connection to the database. For example, the `SYS.MONITOR` table contains information about the number of connections to the database; the number of checkpoints taken; the size of the database; and the amount of memory currently in use. Check the contents of the `SYS.MONITOR` table by executing `SELECT` statements on the columns or by using the `ttIsql monitor` command. For an example of how to use the `ttIsql monitor` command, see "Using the ttIsql Utility" in the *Oracle TimesTen In-Memory Database Operations Guide*.

## Using the query optimizer

The query optimizer is an important tool for performance tuning.

For details about using the query optimizer, see:

- "The TimesTen Query Optimizer" in the *Oracle TimesTen In-Memory Database Operations Guide*
- "Viewing and changing query optimizer plans" in the *Oracle TimesTen In-Memory Database Operations Guide*

If you find that a given query runs more slowly than expected, confirm that the query optimizer has the latest statistics for the tables in your query, as described in "Update query optimizer statistics" on page 2-17. If, after updating your statistics, your query still runs too slowly, it is possible that the TimesTen optimizer is not choosing the optimal query plan to answer that query. Under these circumstances, you can adjust how the optimizer generates a plan by using the `ttOpt` procedures described in "Modifying plan generation" in the *Oracle TimesTen In-Memory Database Operations Guide*. You can also use statement level optimizer hints to influence the optimizer at the statement level. For more information, see "Statement level optimizer hints" in the *Oracle TimesTen In-Memory Database SQL Reference*.

# 2

# Troubleshooting TimesTen Applications and Databases

The following sections provide information to help you diagnose and remedy some problems encountered while using a TimesTen database:

> **Note:** If you are still having problems with your database after following the troubleshooting recommendations in this chapter, contact TimesTen Customer Support.

- Unable to start or stop TimesTen daemon
- No response from TimesTen daemon or subdaemon
- Unable to create shared segment
- Application unable to connect to database in direct mode
- Troubleshooting Client/Server problems
- Application connects or disconnects are slow
- Application is disconnected unexpectedly
- Application is slow
- Application unresponsive, appears hung
- Application unable to find previously created objects
- Troubleshooting OCI and Pro*C/C++ applications
- Running out of a resource
- Duplicate results from a SELECT statement
- Cannot attach PL/SQL shared memory

## Unable to start or stop TimesTen daemon

This section describes what to check if you cannot start or stop the TimesTen main daemon.

| **Possible cause** | **What to do** |
|---|---|
| Privilege is incorrect. | You need the `ADMIN` privilege to start or stop the TimesTen daemon. Ensure that you are using the `ttDaemonAdmin` utility to start the daemon. The output from `ttDaemonAdmin` shows whether you have the correct privilege. |
| Another process is using the TimesTen daemon port. | Use the `ttVersion` utility to verify what port number the TimesTen daemon is expected to use. Use an operating system command like `netstat` to check whether another process is listening on the port. If there is a conflict, either change the port number used by the other process. |
| TimesTen daemon is already running. | Ensure that you are using the `ttDaemonAdmin` utility to start the daemon. The output from `ttDaemonAdmin` shows whether the daemon is already running. |
| There are other problems. | Inspect the user error log produced by the daemon. |

# No response from TimesTen daemon or subdaemon

The following sections describe what to do if one or more of the TimesTen processes appears to be unavailable:

- Check the TimesTen user error log
- Extract a stack trace from the core file

## Check the TimesTen user error log

If you receive an error that indicates the TimesTen subdaemon has stopped and inspect the user error log.

If the TimesTen daemon crashes, it cannot send anything to the user error log, but the subdaemons send a 'main daemon vanished' message to the log before exiting:

```
09:24:13 Err : 4375 ------------------: Main daemon has vanished
```

Restart the daemon. The next connection to each database causes TimesTen to recover from the checkpoint and transaction log files. See "Working with the TimesTen Daemon" in the *Oracle TimesTen In-Memory Database Operations Guide*.

## Extract a stack trace from the core file

If you experience a crash by one of the TimesTen processes on a Linux or UNIX system and have exhausted all of the diagnostic options, check to see if TimesTen has generated a core file. Use the `ttVersion` utility to find the core file. Look for a line in the output that shows a path for the daemon home directory:

```
TimesTen Release (ttuser:40732)
2011-04-04T17:53:04Z
   Instance admin: ttuser
   Instance home directory:
/node1/ttuser/ttcur/TTBuild/linux8664_dbg/install
   Daemon home directory:
/node1/ttuser/ttcur/TTBuild/linux8664_dbg/install/info
```

After locating the core file, attach to the debugger on the system and extract the stack trace from the core file and send the trace results to TimesTen Customer Support.

## Unable to create shared segment

You may receive an error that indicates that a shared segment could not be created:

```
4671: TT14000: TimesTen daemon internal error: Error 28 creating shared segment,
KEY 0x0201f7eb
4671:  -- OS reports too many shared segments in use
4671:  -- Confirm using 'ipcs' and take appropriate action
4671: 18538 ------------------: subdaemon process exited
```

Using the Linux or UNIX `ipcs` command may display information like this:

```
------ Shared Memory Segments --------
 key        shmid      owner     perms      bytes      nattch     status
 0x00000000 1098350592 user1     777        10624      2          dest
 0x00000000 1084817409 user1     777        2439680    2          dest
 0x911fc211 1098383362 user2     666        67108864   1
 0x2814afba 170721285  root      666        1048576    1
```

A status of `dest` means the memory segment is marked to be destroyed. `nattch` shows the number of processes still attached to the memory segment. The `ipcrm` command cannot free the shared memory until the processes detach from the segment or exit. If an application connects to TimesTen and then becomes inactive, nothing can free the shared memory until the user exits or stops the application.

## Application unable to connect to database in direct mode

This section describes what to check if your application cannot connect to a database in direct mode.

| Possible cause | See... |
|---|---|
| There is a mismatch between the release of TimesTen and database. | "Upgrading your database" on page 2-4 |
| User does not have `CREATE SESSION` privilege. | "Privileges to connect to database" on page 2-4 |
| File permissions are incorrect. | "Check file system permissions to access database" on page 2-4 |
| TimesTen daemon or Data Manager service is not running. | "Check that the TimesTen daemon is running" on page 2-4 |
| Incompatible connection attributes or incorrect path name for database set in the DSN. | "Check DSN definition" on page 2-4 |
| There is no available shared memory segment or maximum size of shared memory segment is too small. | "Manage semaphores and shared memory segments" on page 2-4 |
| There is not enough swap space. | "Check available swap space (virtual memory)" on page 2-5 |
| Number of file descriptors is inadequate. | "Increase the number of available file descriptors" on page 2-6 |
| There are other possible causes. | "Using the logs generated by the TimesTen daemon" on page 1-4 |

## Upgrading your database

A database is only guaranteed to be accessible by the same minor release of TimesTen that was used to create the database. When you upgrade the TimesTen software and you would like to use the new release to access a database that was previously created, create a database with the new release. Then use the `ttMigrate` utility to copy the tables, indexes, and table data from the old database to the new one.

See "Upgrades in TimesTen Classic" in the *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide* for details.

## Privileges to connect to database

The user must have the `CREATE SESSION` privilege to connect to the database. If you do not have access, the administrator must use the `GRANT` statement to grant you the `CREATE SESSION` privilege. See "Privileges to connect to the database" in the *Oracle TimesTen In-Memory Database Security Guide*.

## Check file system permissions to access database

A "`permission denied`" error is generated if you attempt to connect to a database and you do not have the proper permissions to access the checkpoint or transaction log files or the directory where those files reside. Check the file system permissions on the files located in the directory specified in the `DataStore` attribute in your DSN.

## Check that the TimesTen daemon is running

If the TimesTen daemon or Data Manager service is not running, an attempt to connect to a database generates TimesTen error 799 "`Unable to connect to daemon; check daemon status.`"

Use the `ttStatus` utility as described in "Check the TimesTen user error log" on page 2-2 to check the status of the TimesTen daemon.

## Check DSN definition

In your DSN description, perform the following:

- Check DSN attributes
- Check path name to database and transaction log directories

### Check DSN attributes

Certain connection options or DSN attribute settings combinations are not compatible. In cases where incompatible settings are used, an error is returned to the application when it attempts to connect to a database.

### Check path name to database and transaction log directories

Confirm that you have specified the correct path names in the `DataStore` and `LogDir` attributes in your DSN. Also confirm that the path names are absolute path names, rather than relative. Otherwise, the path name will be relative to the directory where the application was started.

## Manage semaphores and shared memory segments

An error is generated if you attempt to connect to or create a shared database whose size is larger than the maximum size of shared memory segments configured on your system. Also, an error is generated if the system cannot allocate any more shared memory segments.

On Linux and UNIX systems, use commands similar to the following:

- `ipcs -ma` to check if you have other shared memory segments using up memory, such as Oracle Database instances or other instances of TimesTen.
- `ipcrm` to remove a message queue, semaphore set or shared memory segment identifier. Use `ipcrm` to clean up semaphores or shared memory segments after a faulty TimesTen shutdown, instance crash, daemon crash or other application issues that use shared memory segments and semaphores. Use `-m` to remove a shared memory segment. Use `-s` to remove a semaphore.
- `ps -eafl` to see how much memory is being used by running processes.
- `ulimit -a` to see if there are any limits on the maximum amount of memory one process can address, maximum file size, and the maximum number of open files.

If a shared memory segment is available but is too small to hold your database, use the `ttSize` utility to estimate the amount of memory required for your tables and then check the values of the `PermSize` and `TempSize` attributes to verify the amount of memory established for your database. "Monitoring PermSize and TempSize attributes" in the *Oracle TimesTen In-Memory Database Operations Guide* describes guidelines for setting the size of your permanent and temporary memory regions. If the amount of memory established for your database is too large, reset `PermSize` and `TempSize` to smaller values. See "Check the amount of memory allocated to the database" on page 2-16 for more information. Another option is to increase the maximum size of the shared memory segment, as described below.

If your application is connected to your database and your database is invalidated because of a system or application failure, the database shared segment is not automatically detached from your application. The next time your application tries to use the database, the application will be disconnected from the database and the shared segment will be detached. If your application remains connected to the invalidated database and does not try to use the database, the application will not disconnect from the database and the shared segment will not be detached. To free memory and swap space, make sure you disconnect or terminate all applications that are connected to the invalidated database.

If a database becomes invalidated because of a system or application failure, and you have disconnected all applications from the database, a subsequent connection recovers the database. If recovery fails because you have run out of database space, then reconnect to the database with a larger `PermSize` and `TempSize` value than the ones that are currently in effect.

See "Operating system prerequisites" in the *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide* for more information on how to configure shared memory.

## Check available swap space (virtual memory)

There must be enough swap space to back up shared memory.

On Linux and UNIX systems, use the swap command to check and add virtual memory to your system.

## Increase the number of available file descriptors

Each process connected to a TimesTen database keeps at least one operating system file descriptor open. Additional file descriptors may be opened for a connection when the connection commits or rolls back a transaction. If you receive an error that all file descriptors are in use when attempting to connect to a database, increase the allowable number of file descriptors. See your operating system documentation for limits on file descriptors and information about changing the number of file descriptors. Also see "Limits on number of open files" in the *Oracle TimesTen In-Memory Database Reference*.

# Troubleshooting Client/Server problems

This section includes the following topics:

- Cannot connect to the TimesTen Server
- TimesTen Server failed
- Cannot find Server DSN
- TimesTen Server failed to load DRIVER
- Application times out when accessing TimesTen Server
- TimesTen Client loses connection with TimesTen Server
- Failed to attach to shared memory segment for IPC
- Thread stack overflow when using multiple client connections
- Out of space when DSN specifies new database
- Also consider the topics described in "Application unable to connect to database in direct mode" on page 2-3.

## Cannot connect to the TimesTen Server

You have not correctly identified the system where the TimesTen Server is running.

On a Windows client, select the TimesTen Server in the TimesTen Data Source Setup dialog that is displayed as part of the ODBC Data Source Administrator. To verify the TimesTen Server:

1. On the Windows Desktop, choose **Start** > **Settings** > **Control Panel**.
2. Double click the **ODBC** icon. This opens the ODBC Data Source Administrator.
3. Click the **System DSN** tab. This displays the System Data Sources list.
4. Select the TimesTen Client data source. This opens the TimesTen Client DSN Setup dialog.
5. Click **Servers**. This opens the TimesTen Logical Server List.
6. Select the TimesTen Server from the list. This opens the TimesTen Logical Server Name Setup dialog.
7. Verify that the values for the **Network Address** and **Port Number** are correct. If necessary, change the values.

**Note:** If you typed the host name or network address directly into the Server Name field of the TimesTen Client DSN Setup, the Client tries to connect to the TimesTen Server using the default port.

If the Network Address and Port Number values are correct, the TimesTen Server may not be running. In the *Oracle TimesTen In-Memory Database Operations Guide*, see "Testing connections" for more information about identifying this problem.

On Linux and UNIX, specify the TimesTen Server with the `TTC_Server` connection attribute in the `odbc.ini` file on the client. If the value specified for `TTC_Server` is an actual host name or IP address, the client tries to connect to the TimesTen Server using the default port. In TimesTen, the default port is associated with the TimesTen release number. If the value specified for `TTC_Server` is a logical Server Name, this logical Server Name must be defined in the `ttconnect.ini` file. The `ttconnect.ini` entry for this Server Name must correctly define the host name/IP address and port number on which the TimesTen Server is listening.

If the Network Address and Port Number values are correct, the TimesTen Server may not be running or did not start. See "Testing connections" in the *Oracle TimesTen In-Memory Database Operations Guide* for more information about identifying this problem.

## TimesTen Server failed

Check the server's log file. See "Creating and configuring Client DSNs on Linux and UNIX" and "Managing TimesTen daemon attributes" in the *Oracle TimesTen In-Memory Database Operations Guide* for information on the TimesTen daemon configuration file.

The maximum number of concurrent IPC connections to the Server of a particular TimesTen instance is 24,999. However, TimesTen has a limit of 2043 connections (direct or client/server) to a single DSN.

Client/server users can change the file descriptor limit to support a large number of connections.

## Cannot find Server DSN

On Linux and UNIX, verify that the Server DSN is defined in the `sys.odbc.ini` file on the system running the TimesTen Server.

On Windows, verify that the Server DSN is defined as a System DSN in the ODBC Data Source Administrator on the system running the TimesTen Server. See "Creating and configuring a logical server name on Windows" in the *Oracle TimesTen In-Memory Database Operations Guide*.

## TimesTen Server failed to load DRIVER

This error only occurs on Linux or UNIX systems. Open the `sys.odbc.ini` file on the system running the TimesTen Server and locate the Server DSN you are trying to connect. Verify that the dynamic library specified in the `DRIVER` attribute for the Server DSN exists and is executable.

## Application times out when accessing TimesTen Server

The default `TimeOut` interval is 60 seconds.

To increase this interval on Linux or UNIX, change the value of the `TTC_Timeout` attribute in the `odbc.ini` file.

### TimesTen Client loses connection with TimesTen Server

Check to see if the error was due to the Client timing out. Check the TimesTen Server's log to see why the Server may have severed connection with the Client. Use ping to determine if your network is up or try using `telnet` to connect to the TimesTen Server port number.

### Failed to attach to shared memory segment for IPC

While using shared memory segment (SHM) as IPC, the application may see the following error message from the TimesTen Client ODBC Driver if the application reaches the system-defined per-process file-descriptor-limit.

```
SQLState    = S1000,
Native Error = 0,
Message     = [TimesTen][TimesTen 18.1 CLIENT]Failed to attach to shared memory
segment for IPC. System error: 24
```

This may happen during a connect operation to the Client DSN when the `shmat` system call fails because the application has more open file descriptors than the system-defined per-process file descriptor limit. To correct this problem, you must increase your system-defined per-process file descriptor limit. For more information about file descriptor limits, see "System Limits" in the *Oracle TimesTen In-Memory Database Reference*.

### Thread stack overflow when using multiple client connections

You may receive messages about a segmentation fault that mention a possible thread stack overflow.

If these messages occur, increase the server stack size by one of the following methods:

- Specify the `server_stack_size` option in the `/conf/timesten.conf` file.
- Specify the `ServerStackSize` connection attribute for a specific DSN. This takes precedence over the value in the `/conf/timesten.conf` file.

Increasing the server stack size decreases the number of concurrent connections that can be made before running out of swap space.

See "Working with the TimesTen Client and Server" in the *Oracle TimesTen In-Memory Database Operations Guide*.

### Out of space when DSN specifies new database

You may receive "out of space" messages if you change a DSN to specify a new database while there are existing connections to the original database in a system with multiple client connections.

Close all connections to the original database. This causes a new server process to be created for connections to the database that is now specified in the DSN. Use the `ttStatus` utility to list the connections for the old database. Alternatively, you can restart the server by using the `ttDaemonAdmin` utility with the `-restartServer` option, which resets all client connections on all DSNs in the instance.

## Application connects or disconnects are slow

This section describes what to check if you encounter slow connects and disconnects to a database. First check your `ramPolicy` setting. If the `ramPolicy` setting is `inUse`, and you are the only user, then at first connect, the entire database is loaded into memory. Consider changing `ramPolicy` to either `always` or `manual`. For more information, see "ttAdmin" in the *Oracle TimesTen In-Memory Database Reference*.

| **Possible cause** | **See...** |
|---|---|
| Database is being recovered. | "Check if database is being recovered", below |
| ODBC tracing is enabled. | "Check ODBC tracing", below |
| There are other possible causes. | "API tracing" on page 1-8 |

## Check if database is being recovered

A slow connect may indicate that a TimesTen database is being recovered. This happens only for a first connect.

## Check ODBC tracing

On Windows platforms, if ODBC tracing is enabled, it can slow connect and disconnect speeds. Double-click **ODBC** in the Control Panel to open the ODBC Data Source Administrator. Select the **Tracing** tab and confirm tracing is disabled. See "Using ODBC tracing" on page 1-18.

# Application is disconnected unexpectedly

If an application becomes disconnected from a TimesTen database, one of the following events occurs:

- If there was no outstanding transaction, the connection is cleanly removed by the TimesTen daemon. Other existing connections continue processing as if no problem had occurred.
- If there was an outstanding transaction, the transaction is rolled back and the connection is cleanly removed by the TimesTen daemon. Other existing connections continue processing as if no problem had occurred.

This section describes what to check if your application unexpectedly disconnects from the database.

| **Possible cause** | **See...** |
|---|---|
| There is an internal application error. | "Check for ODBC, JDBC, OCI, Pro*C, and PL/SQL application errors", below |
| There is failure of a concurrent application thread. | "Check for ODBC, JDBC, OCI, Pro*C, and PL/SQL application errors", below<br>"Check the user error log" on page 2-10 |
| If using a client/server connection, the client may have disconnected from the application. | "Troubleshooting Client/Server problems" on page 2-6 |
| There is an error in the TimesTen library. | Contact TimesTen Customer Support. |

## Check for ODBC, JDBC, OCI, Pro*C, and PL/SQL application errors

Check for the following types of errors:

- ODBC errors returned by the `SQLError` function

- JDBC errors returned by the `SQLException` class

TimesTen OCI and Pro*C applications, and those that use PL/SQL, report errors using Oracle database error codes instead of TimesTen error codes. The error messages that accompany the error codes may come from the TimesTen error catalog or the Oracle database error catalog.

By default, TimesTen messages and diagnostic information are stored in:

- A user error log that contains error message information. Generally, these messages contain information on actions you may need to take. The default file is *`timesten_home`*`/diag/tterrors.log`. For more information on modifying the location of the user error log, see "Error, warning, and informational messages" in the *Oracle TimesTen In-Memory Database Operations Guide*.
- A support log containing everything in the user error log plus information of use by TimesTen Customer Support. The default file is *`timesten_home`*`/diag/ttmesg.log`. For more information on modifying the location of the user error log, see "Error, warning, and informational messages" in the *Oracle TimesTen In-Memory Database Operations Guide*.
- An invalidation file containing diagnostic information when TimesTen invalidates a database. This file provides useful troubleshooting information for TimesTen Customer Support. The invalidation file is created and named based on the value specified by the `DataStore` connection attribute. This connection attribute is not a file name. For example on Linux and UNIX systems, if the `DataStore` connection attribute is `/home/ttuser/AdminData`, the actual invalidation file name has a suffix, `.inval`, `/home/ttuser/AdminData.inval`. For more information on the `DataStore` connection attribute, see "DataStore" in the *Oracle TimesTen In-Memory Database Reference*.

It may be helpful to use `ttTraceMon` to generate a level 4 `ERR` trace for the application and review all of the errors messages that are pushed in the TimesTen direct driver. See "ERR tracing" on page 1-12 for details.

## Check the user error log

If a TimesTen application disconnects without returning an ODBC error or any other warning, look through the user error log. See "Using the logs generated by the TimesTen daemon" on page 1-4.

# Application is slow

For details on how to maximize the performance of your application and TimesTen database, see:

- "TimesTen Database Performance Tuning" in the *Oracle TimesTen In-Memory Database Operations Guide*
- "ODBC Application Tuning" in the *Oracle TimesTen In-Memory Database C Developer's Guide*
- "Java Application Tuning" in the *Oracle TimesTen In-Memory Database Java Developer's Guide*

This section describes some issues that impair performance.

| Possible cause | See... |
|---|---|
| Using client/server mode. | "Consider connection mode" on page 2-11 |

| Possible cause | See... |
|---|---|
| Database statistics are outdated. | "Update statistics for your tables" on page 2-11 for information. |
| Committing transactions too frequently. | "Turn off autocommit mode" in the *Oracle TimesTen In-Memory Database Operations Guide*. |
| `DurableCommits` attribute is enabled. | "Use durable commits appropriately" in the *Oracle TimesTen In-Memory Database Operations Guide*. |
| SQL statements used more than once are not prepared. | "Prepare statements in advance" in the *Oracle TimesTen In-Memory Database Operations Guide*. |
| There is a wrong kind of index, too many indexes, or a wrong size for hash index. | "Select the type of index appropriately" in the *Oracle TimesTen In-Memory Database Operations Guide*.<br>"Size hash indexes appropriately" in the *Oracle TimesTen In-Memory Database Operations Guide*. |
| Use of locks is inefficient. | "Verify lock and isolation levels" on page 2-12 for information. |
| Operations are slightly slower when performed on out-of-line columns versus inline. | "Inline and out-of-line columns" in the Oracle TimesTen In-Memory Database Operations Guide. |
| Materialized view is improperly configured. | "Performance implications of materialized views" and "Materialized view tuning" in the *Oracle TimesTen In-Memory Database Operations Guide*. |
| If replication is used, configuration of replication scheme or network environment may be impacting application. | "Poor replication or XLA performance" on page 4-10 for information. |
| If TimesTen Cache is used, TimesTen Cache configuration or environment may be impacting application. | "Poor autorefresh performance" on page 3-23 for information. |
| There are too many table partitions. | "Check partition counts for the tables" on page 2-13 for information. |
| Tracing is unnecessarily enabled for one or more TimesTen components. | "Check trace settings" on page 2-12 for information. |

## Consider connection mode

Client/server connections are slower than direct connections to TimesTen databases. Driver manager connections can also moderately impact performance. The performance overhead imposed by client/server connections can be significant because of the network latencies involved in all communication with the database.

If your application must run on a different system from the one hosting the database, see "Client/Server tuning" in the *Oracle TimesTen In-Memory Database Operations Guide*.

## Update statistics for your tables

The TimesTen query optimizer in general is very good at choosing the most efficient query plan. However, it needs additional information about the tables involved in complex queries in order to choose the best plan. By knowing the number of rows and data distributions of column values for a table, the optimizer has a much better chance of choosing an efficient query plan to access that table.

Before preparing queries that will access a TimesTen table, use the `ttOptUpdateStats` procedure to update the statistics for that table. When updating the statistics for a table, you get the best results if you update statistics on your tables after loading them with data, but before preparing your queries. For example, if you update statistics on a table before populating it with data, then your queries are optimized with the assumption that the tables contain no rows (or very few). If you later populate your tables with millions of rows and then execute the queries, the plans that worked well for the situation where your tables contained few rows may now be very slow.

For more information about updating statistics, see "The TimesTen Query Optimizer" in the *Oracle TimesTen In-Memory Database Operations Guide*.

## Verify lock and isolation levels

The manner in which multiple applications concurrently access the database can have a major impact on performance.

An application can acquire locks on the entire database, individual tables, and individual rows. Additionally, applications can set an isolation level that determines whether they hold read and update locks until their transactions commit or roll back.

Check the `SYS.SYSTEMSTATS` table, the `SYS.MONITOR` table, or use the `ttXactAdmin` utility to detect whether an application is spending time waiting for locks. See "Check for deadlocks and timeouts" on page 2-14 and "Using the ttXactAdmin utility" on page 1-17.

If lock contention is high, you may be able to improve the overall performance of your system by implementing the following:

- Set the `LockLevel` configuration attribute or use the `ttLockLevel` procedure to place locks on rows, rather than on the entire database. Row locking is the default.
- Use the `ttOptSetFlag` procedure to prevent the query optimizer from placing locks on tables. Table locks are sometimes the default, particularly for updates that affect many rows.
- Use read-committed isolation level (`Isolation=1`, the default) for those applications do not require serializable access to the transaction data.

If you see a lot of lock contention, but the above settings are all set to minimize contention, then the contention may be related to the application itself. For example, concurrent threads may be repeatedly accessing the same row. The `ttXactAdmin` utility can sometimes help you detect this sort of contention. Tracing can also be useful in this situation.

For more information about locks and isolation levels, see "Concurrency control through isolation and locking" in the *Oracle TimesTen In-Memory Database Operations Guide*.

## Check trace settings

Use `ttTraceMon -e show` as described in "Using the ttTraceMon utility" on page 1-5 to confirm tracing is off on all TimesTen components. `ERR` should be set to 1; all other components should be set to 0. Trace levels are preserved when a database is reloaded.

On Windows platforms, confirm that ODBC tracing is disabled. Double-click **ODBC** in the Control Panel to open the ODBC Data Source Administrator. Select the **Tracing** tab and confirm tracing is disabled. See "Using ODBC tracing" on page 1-18.

## Check partition counts for the tables

When a table is created, it has one partition. When you use `ALTER TABLE ... ADD COLUMN` to add new columns, a new partition is added to the table. Adding multiple columns with a single `ALTER TABLE ... ADD COLUMN` statement only adds one partition.

There is a limit of 999 partitions per table. Exceeding this number generates error 8204. An extra read for each new partition slightly degrades performance for each of the new partitions. A high partition count should be avoided. On replicated tables that have multiple partitions, additional space is used for each update on the subscriber side, proportional to the number of partitions. This can result in the subscribers using slightly more perm space than the master.

The partition value for each table is tracked in the `SYS16` column of the system table, `SYS.TABLES`. Obtain the partition counts for tables by using the following query:

```
SELECT tblname, sys16 FROM SYS.TABLES;
```

If you discover that a table has too many partitions, do *one* of the following:

- Recreate the table.
- Save and restore the table. Use `ttMigrate -c` to create a migration file. Then restore the table without additional partitions by using `ttMigrate -r -relaxedUpgrade`.

`ALTER TABLE ... DROP COLUMN` does not remove partitions from a table. On replicated systems, all master and subscriber databases must be migrated using the `-relaxedUpgrade` option. Replication does not occur for tables that have different partition structures.

# Application unresponsive, appears hung

This section describes what to check if your application is unresponsive and appears to be hung. First use pstack or equivalent to determine where your application is spending its time.

| **Possible cause** | **See...** |
|---|---|
| There is an internal application error. | "Check for ODBC errors", below |
| Connection attributes set in DSN are inconsistent. | "Consider connection mode" on page 2-11 |
| There is excessive lock contention. | "Check for deadlocks and timeouts", below |

For any possible cause, you can also see the next section, "Check logs and gather trace information".

## Check logs and gather trace information

If your application hangs, check the transaction log by using the `ttXactAdmin` utility. See "Using the ttXactAdmin utility" on page 1-17.

Also check the user error log for errors, as described in "Using the logs generated by the TimesTen daemon" on page 1-4.

You can also generate a trace log to detect the activities on various TimesTen components as described in "Using the ttTraceMon utility" on page 1-5.

## Check for ODBC errors

Check the ODBC errors returned by the `SQLError` function in all applications to determine whether one of them has encountered a problem that caused it to hang. Call `SQLError` after each ODBC call to identify error or warning conditions when they first happen. Examples of `SQLError` usage can be found in the demo programs and in "Retrieving errors and warnings" in the *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

If the problem is repeatable, use `ttTraceMon` to generate a SQL trace to determine where the application is hanging. See "SQL tracing" on page 1-7 for details. In more extreme cases, it may be helpful to generate a level 4 `ERR` trace for the application and review all of the errors messages that are pushed in the TimesTen direct driver. See "ERR tracing" on page 1-12 for details.

## Check for deadlocks and timeouts

If there is no connect problem, a deadlock or timeout may be the problem. The `SYS.SYSTEMSTATS` table and the `SYS.MONITOR` table records information about deadlocks and timeouts. See "Monitoring the TimesTen system tables" on page 1-18 for information on how view the contents of this table. You can also use the `ttXactAdmin` utility to detect the types of locks currently held by uncommitted transactions and the resources on which they are being held.

If a deadlock occurs, the TimesTen subdaemon negotiates the problem by having an application involved in the deadlock generate TimesTen error 6002, "`Lock request denied because of deadlock.`" The error message contains the SQL that the lock holder is running, which can help you diagnose the cause of the deadlock. If your application encounters this error, it should roll back the transaction and then reissue the statements for that transaction. Deadlocks can be caused if your application issues statements in a particular order that results in a circular wait, and can sometimes be prevented by changing the order in which the statements are issued.

An application encounters TimesTen error 6003, "`Lock request denied because of timeout`," if it cannot acquire a lock within the time period defined by the lock timeout interval set by the `LockWait` attribute in the DSN or by the `ttLockWait` procedure in your application. Upon encountering a timeout error, your application can reissue the statement. Keeping transactions short reduces the possibility of lock timeout errors.

In multithreaded applications, a thread that issues requests on different connection handles to the same database may encounter lock conflict with itself. TimesTen resolves these conflicts with lock timeouts.

# Application unable to find previously created objects

This section describes what to check if your application cannot locate previously created tables, indexes, sequences or views in the database.

| **Possible cause** | **See...** |
|---|---|
| Database is temporary. | "Check temporary DSN attribute", below |
| Overwrite attribute is enabled. | "Check Overwrite DSN attribute", below shortly |
| Path name specified in DSN is relative. | "Check path name to database", below |

### Check temporary DSN attribute

Temporary databases (DSN attribute: `Temporary`=1) persist until all connections to the database have been removed. When attempting to access a table in a temporary database and the table does not exist, it is possible that the database in which the table resided in has been dropped.

### Check Overwrite DSN attribute

If the `Overwrite` and `AutoCreate` DSN attributes are enabled and the database already exists, TimesTen drops that database and creates a new one. Any tables that were created in the old database are dropped.

### Check path name to database

To ensure that you are always accessing the same database when connecting to a particular DSN, use an absolute database path name instead of a relative one. For example, if the demo database is in the `datastore` directory, specify:

```
DataStore=/datastore/demo
```

rather than:

```
DataStore=demo
```

In the latter case, the database path name is relative to the directory where the application was started. If you cannot find a table and you are using a relative database path name, it is possible that the database in which the table resides in does exist but the database (checkpoint and log) files are in a different directory than the one that you are accessing.

See "Specifying Data Source Names to identify TimesTen databases" in the *Oracle TimesTen In-Memory Database Operations Guide*.

## Troubleshooting OCI and Pro*C/C++ applications

On Windows, if `NLS_LANG` is not set in the environment, the `NLS_LANG` setting is taken from the registry, `HKEY_LOCAL_MACHINE\Software\ORACLE\NLS_LANG`. If `NLS_LANG` is set to an invalid value or if `NLS_LANG` indicates an unsupported character set, either an OCI connection failed error or an `ORA-12705` error is thrown. For more information on supported character sets, see "Supported character sets" in the *Oracle TimesTen In-Memory Database Reference*.

Refer to the "Globalization support" section in the OCI chapter of the *Oracle TimesTen In-Memory Database C Developer's Guide* for more information on `NLS_LANG`.

## Running out of a resource

This section describes what to check if TimesTen runs out of resources such as memory space, file system space, file descriptors, and semaphores.

| **Symptom** | **See...** |
|---|---|
| Memory consumption seems high. | "Operating system tools and shared memory", below |

| Symptom | See... |
|---|---|
| Running out of memory space. | - "Operating system tools and shared memory", below<br>- "Check the amount of memory allocated to the database" on page 2-16<br>- "Update query optimizer statistics" on page 2-17<br>- "Check memory used by queries" on page 2-17<br>- "Out of memory after fatal crash of the database" on page 2-18 |
| Running out of file system space. | "Check transaction log file use of file system space" on page 2-18<br>"Check if tracing is enabled" on page 2-19 |
| Running out of transaction log space. | "Check transaction log file use of file system space" on page 2-18 |
| Running out of file descriptors. | "Increase the number of available file descriptors" on page 2-6 |
| Running out of semaphores. | "Check the semaphore limit" on page 2-19 |
| Running out of CPU. | Obtain a stack trace and contact TimesTen Customer Support |

## Operating system tools and shared memory

Operating system tools such as `top`, `vmstat`, and `sar` provide statistics about processes and memory usage. The output from these tools can be misleading as an indicator of TimesTen memory consumption because they report shared memory usage for each process but do not report total shared memory usage. Adding together various memory statistics for TimesTen processes overestimates the amount of memory used by TimesTen because shared memory is by definition shared.

## Check the amount of memory allocated to the database

TimesTen uses both permanent and temporary memory regions. The amount of memory allocated for these regions is set by the `PermSize` and `TempSize` attributes in the DSN definition for the database.

When the TimesTen database fills up, it is important to determine whether it is the permanent or the temporary memory region that is filling up. Use the `ttIsql dssize` command to list allocated, in-use, and high water mark sizes for the permanent and temporary memory regions. The `dssize` command selects the following values from `SYS.MONITOR`:

- `PERM_ALLOCATED_SIZE`
- `PERM_IN_USE_SIZE`
- `PERM_IN_USE_HIGH_WATER`
- `TEMP_ALLOCATED_SIZE`
- `TEMP_IN_USE_SIZE`
- `TEMP_IN_USE_HIGH_WATER`

The permanent memory region consists of table and index data, while the temporary memory region consists of internal structures, such as locks, sorting areas, and compiled commands.

Keeping transactions short and making sure there is enough temporary space in the database prevents locks from occupying all of the remaining temporary space. You can also use table locks if transactions are acquiring tens of thousands of row locks.

For tips on how to estimate the size of your database, see "Size your database correctly" in the *Oracle TimesTen In-Memory Database Operations Guide*.

### Permanent memory region filling up

Consider whether you can drop any indexes. You may want to look at query plans to see which indexes are actually used. See "Viewing and changing query optimizer plans" in the *Oracle TimesTen In-Memory Database Operations Guide*. You can also use the `ttRedundantIndexCheck` procedure to discover redundant indexes. The procedure returns suggestions about which indexes to drop.

Use the `ttSize` utility to estimate the amount of memory used by each table in the database. If the amount of data you need to store is too big, you may need to reset the `PermSize` attribute for the database to increase the size of the permanent memory region. Alternatively, you may need to partition your data into several different databases if, for example, you cannot shrink the temporary memory region or create a bigger database because of limits on the memory.

Sometimes when the permanent memory region fills up, copying the data out of the database, deleting all the data, and copying it back in frees up space. This can be done more efficiently by using the `ttMigrate` utility with the `-relaxedUpgrade` option to migrate the data out, destroy and recreate the database, and migrate the data back in.

Finally, you may have to configure the operating system to allow a larger amount of shared memory to be allocated to a process. You may also have to allocate more swap space for virtual memory.

### Temporary memory region filling up

Some commands may be allocating too much space because of out-of-date statistics. See "Update query optimizer statistics", following.

If updating the statistics does not reduce the temporary memory region, disconnect all connections and then reconnect them. Verify that all connections have been disconnected by using the `ttStatus` utility. That frees up all temporary space, but you must reprepare commands.

Diagnose memory usage by queries. See "Check memory used by queries", following shortly.

## Update query optimizer statistics

Make sure you have updated the optimizer statistics with the `ttOptUpdateStats` or `ttOptEstimateStats` procedure. To execute some queries, TimesTen must allocate temporary space. The amount of temporary space required is estimated from statistics about the tables used by the query. Without correct statistics, the temporary space required may be underestimated.

See "Using the query optimizer" on page 1-19.

## Check memory used by queries

You can check the memory that a query uses by observing the high water mark for temporary memory usage. The high water mark represents the largest amount of in-use temporary space used since the high water mark was initialized or reset.

Complete the following tasks:

1. Use the `ttIsql dssize` command to check `TEMP_IN_USE_SIZE` and `TEMP_IN_USE_HIGH_WATER`. Alternatively, you can query the `SYS.SYSTEMSTATS` or the `SYS.MONITOR` table for these values.
2. Call the `ttMonitorHighWaterReset` procedure to reset the `TEMP_IN_USE_HIGH_WATER` to the current value for `TEMP_IN_USE_SIZE`.
3. Execute a query.
4. Use `dssize` to check `TEMP_IN_USE_HIGH_WATER` for peak memory usage for the query.

## Out of memory after fatal crash of the database

Irrecoverable errors, such as errors 846 and 994, invalidate a TimesTen database. However, the database remains in memory, which is only freed after all users have disconnected from the database. If the database is restarted while users are connected to the invalidated database, both old and new instances exist in memory at the same time. In this case, users could receive out-of-memory conditions. To prevent an `"Out of memory"` error, disconnect all active connections at the time of the irrecoverable error before reconnecting. These irrecoverable errors are rare.

## Check transaction log file use of file system space

TimesTen saves a copy of the database in two checkpoint files, which are stored in the directory specified by the `DataStore` attribute. Each checkpoint file can grow on the file system to be equivalent to the size of the database in shared memory. For each permanent database, you must have enough file system space for the two checkpoint files and for transaction log files.

Transaction log files accumulate in the directory specified by the `LogDir` attribute and are only deleted when checkpoints are performed. If the `LogDir` attribute is not specified in the DSN, transaction log files accumulate in the directory specified by the `DataStore` attribute. The maximum size of your transaction log files is set by the `LogFileSize` attribute.

When a file system fills up with TimesTen data, it is most often due to a build-up of transaction log files. Transaction log files are used for numerous purposes in TimesTen, including transaction rollbacks, backups, and replication. It is important to determine which operation is putting a "hold" on the transaction log files, so that appropriate action can be taken to enable the transaction log files to be purged. This can be done by using the `ttLogHolds` built-in procedure. There are six types of log holds. They are discussed in detail below.

- **Long-running transactions** - TimesTen uses the transaction log to roll back transactions. A log hold is placed for the duration of a transaction. Transactions that are active for a long time result in log file building up if the transaction has written at least one log record. (That is, it is not a read-only transaction.) Commit write transactions with reasonable frequency to avoid significant log file build-up. See "Size transactions appropriately" in the *Oracle TimesTen In-Memory Database Operations Guide* for more information on transaction length.
- **Checkpoint** - If a TimesTen application crashes and the database must be recovered, the checkpoint files and transaction log files are used to recover the data. The "most recent" transaction log files are used -- those written since the checkpoint was done. Transaction log files accumulate during the interval between checkpoints. If checkpoints are done very infrequently, a large number of

transaction log files may accumulate, particularly if many changes are made to the database during that interval. See "Checkpoint operations" in the *Oracle TimesTen In-Memory Database Operations Guide*.

- **Replication** -TimesTen replication transmits changes from one database to one or more other databases. It does this by reading the transaction log and sending any relevant changes. If replication goes down, the transaction log files build up. See "Set the replication state of subscribers" in *Oracle TimesTen In-Memory Database Replication Guide* for more information on pausing and restarting or resetting replication.
- **Backup** - TimesTen supports an incremental backup facility that uses transaction log files to augment a backup with changes made since the last backup. Transaction log files accumulate during the interval between incremental backups. To avoid a large log build-up, do incremental backups at relatively frequent intervals. If desired, disable incremental backups and do full backups instead.

  See "Backup, Restore, and Migrate Data in TimesTen Classic" in the *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide* for more information.
- **XLA** - TimesTen's persistent XLA facility reports changes to the database by using transaction log files. Transaction log files are kept until the corresponding transactions have been acknowledged using the `ttXlaAcknowledge` C function. Call `ttXlaAcknowledge` frequently enough to prevent transaction log files building up. See "Retrieving update records from the transaction log" in the *Oracle TimesTen In-Memory Database C Developer's Guide*.
- **XA** - TimesTen's XA support uses transaction log files to resolve distributed transactions. If these transactions are not resolved in a timely manner, transaction log files build up. See "Distributed Transaction Processing: XA" in the *Oracle TimesTen In-Memory Database C Developer's Guide*.

The following attributes are related to file system use.

- The `LogPurge` attribute indicates whether transaction log files that no longer have a hold on them are purged (removed from the file system) or simply archived (renamed). If the `LogPurge` attribute is set to the default value of 0, TimesTen renames transaction log files that it no longer needs by appending the string `.arch` to the name. Once renamed, you must delete the transaction log files manually when they are no longer needed. If transaction log files are not purged, they continue to accumulate space, even when no longer needed by TimesTen.
- The `Preallocate` attribute indicates whether file system space should be reserved for checkpoint files at connect time. This is useful for big databases, to ensure that the file system always has room for the checkpoint files as data is added to the database.

## Check if tracing is enabled

When tracing to a file has been enabled, the file may grow so large that a process attempting an operation may exceed the file limits. Tracing always appends to an existing file.

## Check the semaphore limit

When creating multiple client/server connections to a TimesTen database configured to allow shared memory segment as IPC, you may encounter errors that indicate TimesTen could not create a semaphore.

Semaphore limits are platform-dependent.

## Duplicate results from a SELECT statement

Using read-committed isolation level can lead to duplicates in a result set. A `SELECT` statement selects more or fewer rows than the total number of rows in the table if some rows are added or removed and committed in the range in which the `SELECT` scan is occurring. This may happen when an `UPDATE`, `INSERT` or `DELETE` statement adds or deletes a value from an index and the `SELECT` scan is using this index. This can also happen when an `INSERT` or `DELETE` adds or deletes rows from the table and the `SELECT` operation is using an all-table scan.

Index values are ordered. An `UPDATE` of an index value may delete the old value and insert the new value into a different place. In other words it moves a row from one position in the index to another position. If an index scan sees the same row in both positions, it returns the row twice. This does not happen with a serial scan because table pages are unordered and rows do not need to be moved around for an `UPDATE`. Hence once a scan passes a row, it will not see that same row again.

The only general way to avoid this problem is for the `SELECT` statement to use Serializable isolation. This prevents a concurrent `INSERT`, `DELETE` or `UPDATE` operation. There is no reliable way to avoid this problem with `INSERT` or `DELETE` by forcing the use of an index because these operations affect all indexes. With `UPDATE`, this problem can be avoided by forcing the `SELECT` statement to use an index that is not being updated.

For more information about serializable isolation, see "Concurrency control through isolation and locking" in the *Oracle TimesTen In-Memory Database Operations Guide*.

## Cannot attach PL/SQL shared memory

The `PLSQL_MEMORY_ADDRESS` first connection attribute determines the virtual address at which the PL/SQL shared memory segment is loaded into each process that uses the TimesTen direct drivers. Since each operating system platform has different mappings for its address space, the default values for the PL/SQL address space defined in the `PLSQL_MEMORY_ADDRESS` connection attribute are different for each platform, which avoids conflict with operating system mapped address space.

However, if your application overlaps with the PL/SQL mapped address space, you may receive error 8517 "`Cannot attach PL/SQL shared memory; PLSQL_MEMORY_ADDRESS not valid or already in use.`" In this case, modify the setting for the `PLSQL_MEMORY_ADDRESS` connection attribute to eliminate the overlap. The reasons for receiving error 8517 can be one of the following:

- User allocated memory already uses that address.
- Some shared memory already uses that address.
- A shared library already uses that address.

If an application accesses two or more TimesTen databases at the same time, you must modify the default setting for the `PLSQL_MEMORY_ADDRESS` attribute in all but one of the TimesTen databases, since the default settings would map the PL/SQL memory address to the same address for all TimesTen databases.

# 3

# Troubleshooting TimesTen Application-Tier Database Cache

The following sections in this chapter describe how to troubleshoot problems you may encounter when using TimesTen Application-Tier Database Cache (TimesTen Cache):

- Unable to create a cache group
- Unable to start or stop the cache agent
- Unable to resolve Oracle Service Name
- Unable to resolve connect identifier
- Incompatible Oracle Database Server and Client versions
- Unable to validate the Oracle database user name and password
- OCI initialization failed
- Unsupported data type mapping
- Null constraint does not match Oracle Database
- DDL on cached Oracle database tables may cause cache group operations to fail
- Changes not visible after updating object in cache group
- Loading or refreshing fails
- Monitoring autorefresh cache groups
- Optimize performance for TimesTen Cache
- Avoid performance and memory problems for large batch jobs on Oracle database tables
- Autorefresh not refreshing cache at the specified interval
- Incremental autorefresh not progressing
- Incremental autorefresh becomes full autorefresh
- Poor autorefresh performance
- Declaring NOVALIDATE on constraints causes cache group creation failure
- AWR report showing lock contention with DBMS_LOCK

If you are having problems with an AWT cache group, see Chapter 5, "Troubleshooting AWT Cache Groups".

---

**Note:** Error log messages from the cache agent daemon are designated by `CAC` in the message, as shown in several examples in this chapter. These messages are enabled unless you disable the `CACHE` component of the `ttDaemonLog` utility. See "ttDaemonLog" in *Oracle TimesTen In-Memory Database Reference* for information about `ttDaemonLog` options for Windows and Linux or UNIX systems.

---

## Unable to create a cache group

This section describes problems you might encounter when executing the `CREATE CACHE GROUP` statement.

| Possible cause | What to do |
|---|---|
| User does not have the correct Oracle database privileges to create the cache group type. | See "Check the Oracle database privileges" on page 3-5. |
| User has insufficient access to the database. | You must have `CACHE_MANAGER` privilege to create a cache group. |
| The internal/external user does not match the Oracle database user. | The TimesTen user name must be the same as the Oracle database user name. |
| Cannot connect to the Oracle database. | See:<br>■ "Unable to resolve Oracle Service Name" on page 3-3<br>■ "Unable to resolve connect identifier" on page 3-4<br>■ "Unable to validate the Oracle database user name and password" on page 3-5<br>■ "Incompatible Oracle Database Server and Client versions" on page 3-4<br>■ Check whether Oracle Database must be restarted.<br>Check the network status. |
| Cache administration user ID or password is not set (when trying to create AWT or autorefresh cache groups). | See "Set the cache administration user name and password" on page 3-6. |
| Data type mapping is unsupported. | See "Unsupported data type mapping" on page 3-8. |
| Nullability setting in the Oracle database is different. | See "Null constraint does not match Oracle Database" on page 3-8. |
| Primary key in root table is not specified. | The root table of a cache group must have a primary key. See "Defining Cache Groups" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*. |

## Unable to start or stop the cache agent

This section describes problems you might encounter when starting or stopping the cache agent.

| Possible cause | What to do |
|---|---|
| Cache agent is already running. | See "Check status of the cache agent", below. |

| Possible cause | What to do |
|---|---|
| Cannot locate Oracle Database libraries. | ■ See "Check status of TNS listener and Oracle Database Server" on page 3-5.<br>■ Check the permissions on the libraries. |
| `ORACLE_HOME` is invalid. | See "Check ORACLE_HOME environment variable", below. |
| Privileges are insufficient. | You must have `CACHE_MANAGER` privilege to start or stop the cache agent. |
| `OracleNetServiceName` is incorrect. | Ensure that the `OracleNetServiceName` set in your DSN definition matches the Oracle Service Name for the Oracle Database instance that contains the tables to cache in TimesTen. |

## Check status of the cache agent

Check the status of the cache agent by using the `ttStatus` utility as described in "Using the ttStatus utility" on page 1-2 to check the status of the cache agent.

If the cache agent is not running, start it as described in "Starting the cache agent" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*. If attempts to start the cache agent fail, then investigate the possible causes and restart the system before attempting to start the cache agent.

## Check ORACLE_HOME environment variable

On Linux and UNIX systems, check that the `ORACLE_HOME` environment variable is set correctly for the shell from which you are starting the cache agent and the TimesTen daemon.

See "Environment variables" in *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide* for more information.

## Check NLS environment variables

NLS environment variables are set in the environment where the TimesTen application is running. Check to see if TimesTen is using the NLS environment variables.

# Unable to resolve Oracle Service Name

If you receive error `ORA-12514` indicating "`could not resolve service name`":

- Use the Oracle Database `TNSPING` utility to verify that the service can be reached.
- Ensure that the `OracleNetServiceName` set in your DSN definition matches the Oracle Service Name for the Oracle Database instance that contains the tables to cache in TimesTen.
- Ensure that there is a service name defined. If it is a Windows Oracle client, use Oracle Net Configuration Assistant to configure a service name. In Oracle Net Configuration Assistant, navigate to Oracle Net Configuration -> Local -> Service Naming, select your Oracle Database server and confirm that there is a service name or a SID that identifies the Oracle Database server. If you add or modify a service name, you may need to restart.

  Check the cache administration user name and password on the Oracle database with SQL*Plus to make sure this service name works. For example:

```
%sqlplus cache_admin_user/cache_admin_pwd@OracleHost
```

*`cache_admin_user`* is the cache administration user name, *`cache_admin_pwd`* is the cache administration user password, and *`OracleHost`* is the `OracleNetServiceName` specified in your DSN definition.

> **Note:** Your cache administration user may be different from your regular Oracle database user. See "Create the Oracle database users" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

- Ensure that there is only one copy of `tnsnames.ora` on your TimesTen system. Also check the permission on `tnsnames.ora`.
- If you run TimesTen on a Linux or UNIX system, check that the `ORACLE_HOME` environment variable is correctly defined. For example:

  ```
  ORACLE_HOME=/products/oracle11g
  ```

- Check the Oracle Database client and server versions. See "Incompatible Oracle Database Server and Client versions", following shortly.

## Unable to resolve connect identifier

You may receive `ORA-12154` "`TNS:could not resolve the connect identifier specified`" when you try to connect to a a database.

This can occur when you are trying to use TimesTen Cache and Oracle Database on the same system and the `TNS_ADMIN` environment variable does not point to the proper `tnsnames.ora` file for Oracle Database. For example, you may have several instances of the Oracle Database running on a laptop.

In a production environment, you typically have TimesTen and Oracle Database running on different systems. In this case, do not reset the `TNS_ADMIN` environment variable to point to a `tnsnames.ora` file on the system where TimesTen is running. The Oracle Database client uses the `TNS_ADMIN` setting to resolve the connection, but the TimesTen main daemon, the cache agent, the Web server, and the replication agent are unaware of the `TNS_ADMIN` setting. TimesTen Cache cannot operate properly when the Oracle Database client and TimesTen use different `tnsnames.ora` files.

On Windows, set the `TNS_ADMIN` environment variable as follows:

1. Right-click My Computer and choose Properties.
2. On the Advanced tab, choose Environment Variables.
3. Add or edit `TNS_ADMIN` as a system environment variable so that it points to the directory that contains the `tnsnames.ora` file that you want to use. You can include other `tnsnames.ora` files with the `INAME` command inside the `tnsnames.ora` file.

## Incompatible Oracle Database Server and Client versions

If you receive connection timeout errors such as `ORA-12170` or `ORA-12535`, or if you receive `ORA-03134` (server version not supported), verify that you are using an Oracle Database client and Oracle Database server whose versions are compatible.

# Unable to validate the Oracle database user name and password

This section describes problems you might encounter when using the Oracle database user name and password.

| Possible cause | See... |
|---|---|
| The library environment variable is not set correctly. | "Check library path environment variable", below |
| Oracle Database processes are not running. | "Check status of TNS listener and Oracle Database Server", below |
| User does not have the correct Oracle database privileges. | "Check the Oracle database privileges", below |
| DSN is incorrectly configured. | "Check DSN definition" on page 3-5 |
| There are problems with cache administration user ID or password. | "Set the cache administration user name and password" on page 3-6 |
| User and system environments are inconsistent. | "Check user and system environment" on page 3-6 |
| Dynamic libraries are not loading. | "Verify the loaded dynamic libraries" on page 3-7 |

## Check library path environment variable

Check the library path environment variable on your platform.

| On this platform... | Check this variable... |
|---|---|
| Linux or UNIX | `LD_LIBRARY_PATH` |
| Windows | `PATH` |

## Check status of TNS listener and Oracle Database Server

Try to connect to the Oracle database by using SQL*Plus or use Oracle Enterprise Manager to verify the status.

## Check the Oracle database privileges

From an Oracle SQL*Plus command prompt, list the current Oracle database privileges granted to you by entering:

```
SELECT * FROM SESSION_ROLES;
SELECT * FROM SESSION_PRIVS;
```

Compare the privileges listed against the required privileges for the various TimesTen Cache operations that are specified in "Grant privileges to the Oracle database users" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*. Contact your Oracle Database Administrator if you require additional privileges.

## Check DSN definition

- Confirm you have correctly set the DSN attributes, such as in the example "DSN for a TimesTen database that caches data from an Oracle database" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.
- Confirm that the DSN definition for TimesTen Cache is a system DSN.

- Confirm that the DSN for TimesTen Cache is defined only once.
- Confirm the Oracle database user name and password. Use SQLPlus and connect to the Oracle database using the same `OracleNetServiceName` and `OraclePWD` used in your DSN definition to confirm they are correct.

## Reboot TimesTen machine

If the Oracle Database client was installed and the system has not been restarted, then the TimesTen daemon is still running under the "old" environment before the Oracle Database client install. Restart your system so the TimesTen can start under the "new" environment.

## Set the cache administration user name and password

The cache administration user name and password must be set only once in a TimesTen database. However, they must be changed if the TimesTen database is destroyed and recreated or if the cache administration user name is dropped and recreated in the Oracle database.

The cache administration user name and password cannot be changed if the cache agent is running on the TimesTen database or there are cache groups in the database. The cache groups must be dropped before you can change the cache administration user name and password. You must also stop the cache agent before you change the cache administration user name and password, and then restart the cache agent after the user name and password have been changed.

From a `ttIsql` session, connect to the database as the cache manager user and call the `ttCacheUidPwdSet` built-in procedure to set the Oracle database cache administration user name and password, as follows:

```
Command> call ttCacheUidPwdSet('cacheuser','oracle');
```

If it returns an error, check the Oracle Database ID, the cache administration user ID and cache administration password. In addition, check whether the Oracle Database instance is running.

You can also set the user name and password by executing the `ttAdmin -cacheUidPwdSet` utility command as a TimesTen external user with the `CACHE_MANAGER` privilege:

```
% ttAdmin -cacheUidPwdSet -cacheUid cacheuser -cachePwd oracle cachealone1
```

If you do not specify the `-cachePwd` option, the `ttAdmin` utility prompts for the cache administration user's password. For more information about the utility, see "ttAdmin" in *Oracle TimesTen In-Memory Database Reference*.

## Check user and system environment

Test to see if the problem is due to differences in user and system environment. This procedure requires two session windows (Command Prompt windows in Windows or shell windows in Linux or UNIX).

1. Stop the TimesTen daemon.
2. In one session window, start the Timesten daemon as a regular user.

   On Windows:

   ```
   % timesten_home/install/srv/ttsrv181.exe -d -verbose
   ```

On Linux or UNIX:

```
% timesten_home/install/srv/timestend -d verbose
```

Some messages will flash by, and then it goes into a wait state.

3. In another session window, try to restart the cache agent.
4. If Step 3 succeeds, then use Ctrl-C on Windows or the `kill` command on Linux or UNIX to stop the TimesTen daemon you started for the other session in Step 2.
5. Compare the user environment and system environment. For example, do both user and system see the same copy of `oci.dll`? Are there any differences in the path name to the `oci.dll` library between the user and system environments?
6. If you detect differences, make the necessary modifications.
7. Restart the system and restart the TimesTen daemon.

## Verify the loaded dynamic libraries

If you run on a Windows system with Visual C++ installed, verify the loaded dynamic libraries. This works only if you can start the cache agent without autorefresh:

1. Make sure TimesTen is started.
2. Start the cache agent without autorefresh.

```
Command> call ttCacheStart;
Command> create cache group cg1 from t1(c1 int not null primary key);
```

3. Open the Windows Task Manager, find process `ttora181.exe` and highlight it. Right-click on it and select Debug. This brings you into Visual C++ and you should see the loaded DLL in the debug window, as described in "Unable to resolve Oracle Service Name" on page 3-3.
4. Load the cache group to force an cache connection from the cache agent:

```
Command> load cache group cg1 commit every 100 rows;
```

5. Review the loaded DLL in your debug window.

# OCI initialization failed

Error 5105, "`OCI initialization failed`," may occur when an operation requires contact with the Oracle database. For example, the error might occur in the following situations:

- Starting the cache agent
- Setting the cache administration user ID or password
- Entering a SQL statement in TimesTen when autocommit=0 and PassThrough=3

Error 5105 contains additional information about its cause:

- OCI cannot find an Oracle database library. See "Check library path environment variable" on page 3-5 and check the permissions on the library specified in the error message.
- `ORACLE_HOME` is invalid. See "Check ORACLE_HOME environment variable" on page 3-3.

- NLS environment variables are set in the environment where the TimesTen application is running. Check to see if TimesTen is using the NLS environment variables.

## Unsupported data type mapping

When you try to create a cache group, you may receive the following error:

```
5115: Unsupported type mapping for column name
```

For example, table *tab* on the Oracle database can be described as follows:

```
COL1     NUMBER(38) NOT NULL
COL2     NUMBER(38)
```

Try to create the cache group as follows:

```
CREATE CACHE GROUP cg FROM tab(col1 CHAR(10) NOT NULL PRIMARY KEY);
```

Error 5119 is displayed and the cache group is not created because the statement attempts to map a column of `NUMBER` data type to a column of `CHAR` data type.

See "Data type mappings allowed for key columns" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

## Null constraint does not match Oracle Database

When you try to create a cache group, you may receive the following warning:

```
Warning 5119: Column name has different nullability setting in Oracle
```

For example, table `tab` on the Oracle database can be described as follows:

```
COL1     NUMBER(38) NOT NULL
COL2     NUMBER(38)
```

Try to create the cache group as follows:

```
CREATE CACHE GROUP cg
    FROM tab(col1 INTEGER NOT NULL PRIMARY KEY, col2 INTEGER NOT NULL);
```

Warning 5119 is displayed because `col2` on the Oracle database does not have a `NULL` constraint, but `col2` in the cache group is defined as `NOT NULL`.

## DDL on cached Oracle database tables may cause cache group operations to fail

DDL operations that are performed on an Oracle database table that is being cached in TimesTen may cause a failure on the cache group. For example, the user drops a column on the Oracle database table that is being cached in TimesTen. When the cache group is propagated or flushed, TimesTen will update the column that no longer exists in the Oracle database table. When the cache group loads or refreshes, then TimesTen attempts to retrieve data from the column that has been dropped.

The following cache group operations may fail:

- Autorefresh does not occur.
- AWT cache group operations are not propagated or refreshed to or from the Oracle database.

- Cache group load or propagate fails.

If you suspect the cache group operations are not working properly because of a DDL operation on the Oracle database base table, then use DDL tracking to diagnose the issue. DDL tracking saves the change history for all the cached Oracle database tables. The SQL statement and when it was executed are each written to a TimesTen table in the cache administrator user schema on the Oracle database.

For more information on how to create the DDL tracking objects and how to enable DDL tracking for the base table within the Oracle database, see "Tracking DDL statements issued on cached Oracle Database tables" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*. For details on the built-in procedure used for enabling or disabling DDL tracking, see "ttCacheDDLTrackingConfig" in *Oracle TimesTen In-Memory Database Reference*.

## Changes not visible after updating object in cache group

If you modify an object in a cache group and then the changes do not appear on a subsequent SQL statement, then one of the following may have occurred:

- The object was dropped from the Oracle database or was somehow damaged.
- Oracle database was restored or recovered to a time before the object was created.
- The Oracle database was down.
- The user modified the `OracleNetServiceName` DSN or connection attribute after creating the cache group, which points to an Oracle database other than the one that the cache group was created upon.

For example, if the user creates an AWT cache group. Then, the user added rows to a table. When the user performs a `SELECT * FROM` the table, the rows did not appear. The `ttmesg.log` error file does not display an error that the Oracle database is not available. Instead, it displays the following messages:

```
12:09:02.10 Err : REP: 29934: CACHE1:meta.c(904): TT5221: TT5221: Oracle syntax
error in OCIStmtExecute(): ORA-00942: table or view does not exist rc = -1 --
file "bdbStmt.c", lineno 1535, procedure "getOraOutTypesNLengths()"
12:09:02.27 Err : REP: 29934: CACHE1:receiver.c(1978): TT5250: Awt Initialization
Failure. Could not compile meta data sql.
12:09:02.27 Warn: REP: 29934: CACHE1:transmitter.c(6505): TT16060: Failed to read
data from the network. select() timed out
```

To recover, perform the following:

1. Stop all updates to the cache group.
2. If you are using an AWT cache group, then flush the cache group.
3. Recreate the cache group with the drop and create.

## Loading or refreshing fails

If the `LOAD CACHE GROUP` or `REFRESH CACHE GROUP` statement fails when you specify `COMMIT EVERY` *n* `ROWS` and *n* is greater than 0, the contents of the target cache group could be in an inconsistent state. Some cache instances may be partially loaded.

Unload the cache group and then load it again. In some situations, it may be easier to drop and recreate the cache group.

# Monitoring autorefresh cache groups

This section includes the following topics:

- Using the ttCacheAutorefreshStatsGet procedure
- Displaying information from the change log tables
- Understanding messages about autorefresh in the support log
- Diagnosing autorefresh failure
- Diagnosing autorefresh performance problems

## Using the ttCacheAutorefreshStatsGet procedure

The `ttCacheAutorefreshStatsGet` procedure returns information about the last ten autorefresh operations on a specified cache group.

The `ttCacheAutorefreshStatsGet` procedure returns information only when the cache agent is running and the autorefresh state is `ON` or `PAUSED`. All of the return fields are set to 0 when the cache agent is restarted or the autorefresh state is changed to `OFF`.

***Example 3–1 Calling ttCacheAutorefreshStatsGet***

This example uses `testcache`, which is a READONLY cache group with one table and an incremental autorefresh interval of 10 seconds.

```
Command> call ttcacheautorefreshstatsget('user1','testcache');

< 1164260, 2007-07-23 15:43:52.000000, 850280, 44, 0, 75464, 528255, 75464, 310, 110, 6800,
1890912, 12439795, 1890912, 160020, InProgress >
< 1164260, 2007-07-23 15:43:33.000000, 831700, 43, 13550, 108544, 759808, 108544, 1030, 230,
12290, 1815448, 11911540, 1815448, 160020, Complete >
< 1164260, 2007-07-23 15:43:12.000000, 810230, 42, 17040, 115712, 809984, 115712, 610, 330,
16090, 1706904, 11151732, 1706904, 146470, Complete >
< 1164260, 2007-07-23 15:42:52.000000, 790190, 41, 14300, 94208, 659456, 94208,560, 320,
13410, 1591192, 10341748, 1591192, 129430, Complete >
< 1164260, 2007-07-23 15:42:32.000000, 770180, 40, 12080, 99328, 695296, 99328,450, 290,
11340, 1496984, 9682292, 1496984, 115130, Complete >
< 1164260, 2007-07-23 15:42:12.000000, 750130, 39, 10380, 86016, 598368, 86016,430, 230,
9720, 1397656, 8986996, 1397656, 103050, Complete >
< 1164260, 2007-07-23 15:41:52.000000, 730130, 38, 13530, 112640, 700768, 112640, 530, 220,
12780, 1311640, 8388628, 1311640, 92670, Complete >
< 1164260, 2007-07-23 15:41:32.000000, 710120, 37, 9370, 56320, 326810, 56320, 310, 160,
8900, 1199000, 7687860, 1199000, 79140, Complete >
< 1164260, 2007-07-23 15:41:22.000000, 700120, 36, 2120, 10240, 50330, 10240, 50, 200, 1870,
1142680, 7361050, 1142680, 69770, Complete >
< 1164260, 2007-07-23 15:41:12.000000, 690110, 35, 0, 0, 0, 0, 0, 0, 0, 1132440, 7310720,
1132440, 67650, Complete >
10 rows found.
```

Table 3–1 describes the results from the first row of output.

***Table 3–1 ttCacheAutorefreshStatsGet results from last autorefresh operation***

| Result | Field name | Description |
|---|---|---|
| `1164260` | *`cgId`* | Cache group ID |
| `2007-07-23 15:43:52.000000` | *`startTimestamp`* | Timestamp when autorefresh started for this interval |

**Table 3–1 (Cont.) ttCacheAutorefreshStatsGet results from last autorefresh operation**

| Result | Field name | Description |
|---|---|---|
| `850280` | *`cacheAgentUpTime`* | Number of cache agent clock ticks in milliseconds at the time the autorefresh transaction started for this interval<br>This value is cumulative and is reset when the cache agent process starts. |
| `44` | *`autorefNumber`* | Autorefresh number |
| `0` | *`autorefDuration`* | Number of milliseconds spent in this autorefresh operation<br>It is zero because the operation is in progress. |
| `75464` | *`autorefNumRows`* | Number of rows autorefreshed in this autorefresh operation<br>This would include all rows in the root table and child tables if the cache group had child tables.<br>**Note**: This information is not provided for full autorefresh. |
| `528255` | *`numOracleBytes`* | Number of bytes transferred from the Oracle database in this autorefresh operation<br>**Note**: This information is not provided for full autorefresh. |
| `75464` | *`autorefNumRootTblRows`* | Number of root table rows autorefreshed in this autorefresh operation |
| `310` | *`autorefQueryExecDuration`* | Duration in milliseconds for the autorefresh query to execute on the Oracle database<br>**Note**: This information is not provided for full autorefresh. |
| `110` | *`autorefQueryFetchDuration`* | Duration in milliseconds for the autorefresh query to fetch rows from the Oracle database<br>**Note**: This information is not provided for full autorefresh. |
| `6800` | *`autorefTtApplyDuration`* | Duration in milliseconds for TimesTen to apply the updated rows to the cache group<br>**Note**: This information is not provided for full autorefresh. |
| `1890912` | *`totalNumRows`* | Total number of rows autorefreshed since the cache agent started<br>**Note**: This information is not provided for full autorefresh. |
| `12439795` | *`totalNumOracleBytes`* | The total number of bytes transferred from the Oracle database since the cache agent started<br>**Note**: This information is not provided for full autorefresh. |
| `1890912` | *`totalNumRootTblRows`* | Total number of root table rows autorefreshed since the cache agent started |
| `160020` | *`totalDuration`* | Total autorefresh duration in milliseconds since the cache agent started |

*Table 3–1 (Cont.) ttCacheAutorefreshStatsGet results from last autorefresh operation*

| Result | Field name | Description |
|---|---|---|
| `InProgress` | *autorefreshStatus* | Status<br>Status can also be `Complete` or `Failed`. |

Note that the total number of autorefreshed rows (1890912) is the same as the total number of autorefreshed root table rows in this example because there are no child tables.

The number of autorefreshed rows in TimesTen does not necessarily reflect the number of rows updated on the Oracle database. The Oracle database updates may be applied in TimesTen more than once, or multiple Oracle database updates on the same row may be applied as one update in TimesTen.

## Displaying information from the change log tables

TimesTen provides the `cacheInfo` SQL script that gathers information from the change log tables that exist on the Oracle database for autorefresh cache groups. See "Managing a caching environment with Oracle Database objects" in the *Oracle TimesTen Application-Tier Database Cache User's Guide* for additional information about change log tables.

Run the script as the cache administration user on the Oracle database using SQL*Plus. If you run the script as a different user, it reports that the change log tables do not exist.

The script is in the following location:

```
timesten_home/install/oraclescripts/cacheInfo.sql
```

The `cacheInfo` script displays the following information for each cached table:

```
% cd timesten_home/install/oraclescripts
% sqlplus cacheuser/oracle
SQL> @cacheInfo
*************Autorefresh Objects Information ***************
Host name: sys1
Timesten datastore name: /users/OracleCache/alone1
Cache table name: ORATT.ORDERS
Change log table name: tt_06_69245_L
Number of rows in change log table: 100000
Maximum logseq on the change log table: 38
Timesten has autorefreshed updates up to logseq: 38
Number of updates waiting to be autorefreshed: 0
Number of updates that has not been marked with a valid logseq: 0
****************************
```

The information returned for each change log table includes the name of the change log table, the name of its corresponding TimesTen cache table, the number of rows in the change log table, and the number of updates in the change log table that have not been automatically refreshed into the cache table.

The log sequence number (`logseq`) acts as a marker for the autorefresh operation.

## Understanding messages about autorefresh in the support log

The support log contains messages that show the progress of autorefresh. For example, `testcache` is a readonly cache group with an autorefresh interval of 10 seconds (10,000 milliseconds).

The support log shows when autorefresh starts:

```
15:43:33.96 Info: CAC: 5264: TT47118-5264-5676-refresh03918: Starting autorefresh
number 43 for interval 10000ms
```

The message includes the following information:

- Message number: TT47118
- Timestamp (`15:43:33.96`)
- Cache agent process ID (`5264`)
- Thread ID (`5676`)

You can look up the message number in the *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*. For example, in the log message above, TT47118 is error message number 47118.

The thread ID is important because autorefresh numbers are unique only for a specific interval. Always check both the thread ID and the autorefresh number when you are tracking a specific autorefresh operation.

The support log also contains a longer message that reports information similar to the `ttCacheAutorefreshStatsGet` procedure. 108544 rows were updated in this autorefresh interval, and 1815448 rows have been updated since the cache agent was started. Note that the total number of rows and the total number of root table rows are the same in this message because there is only one table in the cache group. `Number` refers to the autorefresh number. All times are expressed in milliseconds.

```
15:43:51.81 Info: CAC: 5264: TT47087-5264-5676-refresh04387: Cache agent
refreshed cache group USER1.TESTCACHE: Number - 43, Duration - 13550, NumRows -
108544, NumRootTblRows - 108544, NumOracleBytes - 759808, queryExecDuration -
230, queryFetchDuration - 1030, ttApplyDuration - 12290, totalNumRows - 1815448,
totalNumRootTblRows - 1815448, totalNumOracleBytes - 11911540, totalDuration -
160020
```

Additional messages show that the autorefresh operation completes successfully:

```
15:43:51.81 Info: CAC: 5264: TT47119-5264-5676-refresh04449: Autorefresh
number 43 finished for interval 10000ms successfully
15:43:51.81 Info: CAC: 5264: TT47119-5264-5676-fresher01619: Autorefresh number
43 succeeded for interval 10000 milliseconds
```

Inspect the timestamps to determine whether autorefresh is progressing as expected.

See "Managing TimesTen daemon attributes" in the *Oracle TimesTen In-Memory Database Operations Guide* for information about setting the support log location.

## Diagnosing autorefresh failure

If `ttCacheAutorefreshStatsGet` shows that the status of an autorefresh operation is **Failed**, check the support log for messages related to the autorefresh operation with number the number shown in the `ttCacheAutorefreshStatsGet` output. Look for errors that occurred after the autorefresh operation started.

***Example 3–2 ttCacheAutorefreshStatsGet output shows autorefresh failure***

This row of output from `ttCacheAutorefreshStatsGet` shows a failed autorefresh operation.

```
< 1164260, 2007-08-01 14:56:36.000000, 959350, 9, 0, 0, 0, 0, 0, 0, 0, 1, 7,
 1, 50, Failed >
```

The autorefresh number is 9.

The support log shows the start message for autorefresh number 9:

```
14:56:36.10 Info: CAC: 5988: TT47118-5988-4724-refresh03926: Starting
autorefresh number 9 for interval 15000ms
```

The thread ID for autorefresh number 9 is 4724. Look for error messages with this thread ID.

The following messages appear in the support log:

```
14:56:36.10 Info: CAC: 5988: TT47117-5988-4724-refresh03953: Autorefresh thread
for interval 15000ms is connected to instance inst1 on host host1. Server handle
231976252
14:56:36.12 Err : CAC: 5988: TT40018-5988-4724-refresh07567: TimesTen error
code:5901, msg The Oracle refresh log table, "USER2"."TT_06_81799_L", for base
table, USER2.READTAB2, cannot be found.
14:56:36.12 Info: CAC: 5988: TT47055-5988-4724-refresh05559: Autorefresh rolled
back.
14:56:36.12 Info: CAC: 5988: TT47119-5988-4724-refresh04458: Autorefresh number
9 finished for interval 15000ms with error.
14:56:36.12 Err : CAC: 5988: TT40035-5988-4724-fresher01606: Autorefresh number
9 failed for cache groups with interval 15000 ms after 10 retries.
```

The error message for thread ID 4724 shows that the change log table, `TT_06_81799_L`, is missing. The introduction to "Autorefresh not refreshing cache at the specified interval" on page 3-17 has a table entry that describes what to do in this situation.

## Diagnosing autorefresh performance problems

You can use the `ttTraceMon` utility to diagnose autorefresh performance problems. See "AUTOREFRESH tracing" on page 1-14.

TimesTen tracing severely impacts application performance and consumes a great deal of file system space if trace output is directed to a file. When you are finished, reset tracing to the default values.

It is also suggested to compare Automated Workload Repository (AWR) Reports when autorefresh is working and when it is not.

# Optimize performance for TimesTen Cache

The following recommendations may optimize performance for the TimesTen Cache:

> **Note:** Each of these suggestions involve performance tradeoffs, which may not always be beneficial for optimal use. Consider and test each performance suggestion for your own configured environment.

- Pin the TimesTen Cache meta tables and cache group base tables in the SGA. by executing the `ALTER TABLE` *`table_name`* `CACHE` statement to indicate to the Oracle database that these tables should be stored in the keep portion of the SGA buffer cache. Pinning TimesTen Cache tables in the SGA increases the probability that any given data block needed for a TimesTen Cache refresh operation will be available in the SGA when the refresh is performed and will not force a file system read. This minimizes physical file system reads executed during TimesTen cache refresh operations.

- Pin TimesTen Cache triggers into the shared pool using the `dbms_shared_pool.keep` procedure. Pinning triggers into the shared pool for applications where updates to the cache group base tables are infrequent keeps the trigger from having to be reloaded and reparsed. This is not necessary for highly volatile tables where the trigger will be executed frequently and will remain in the shared pool under any circumstances.
- Enable parallel query. For very large base tables with 10 million rows or more, consider using the Oracle database parallel query facility. The primary join query between the log table and the base table is the kind of query which the Oracle database parallel query is designed to handle. When parallel processing is enabled, the parallel query optimizer generates a query plan that enables the original query to be broken into sections to be worked concurrently by different parallel query worker processes. When using parallel query, users should assign a default degree of parallelism of (`2*N`) to the cache group base table, where "N" is the number of CPUs on the system. Then, experiment to understand what level of parallelism works best for their environment. Experiment with different table structures for base tables, as follows.
  - Use a standard heap table with default degree of parallelism assigned during table creation or by use of the `ALTER TABLE PARALLEL` command. Build an N-partition primary key index against the table.
  - Use an N-way partitioned table structure with partition range key based either on the table primary key or, in the case of a concatenated primary key, the high-order column of the primary key. The number of partitions should be set to the degree of parallelism. Use a local primary key index with the same number of partitions.
  - Use an N-way hashed partition structure using the primary key as the hash key, a local partitioned primary key index, and both index and table partitions equal to the degree of parallelism. The log table should not be partitioned, as the cardinalities of the log table should never be large enough that a partitioned log table would have any performance benefit. Further, given the continuously increasing value of the log table primary key column, range partitions cannot be used.

## Avoid performance and memory problems for large batch jobs on Oracle database tables

Customers sometimes run large batch jobs at month-end or year-end on the Oracle database tables that are cached in read-only cache groups with incremental autorefresh. This can cause performance and memory problems for autorefresh operations and replication unless preventative steps are taken.

The tasks described in this section are supported for these components in an TimesTen Cache configuration:

- Active standby pair replication
- Active standby pair with a return service specified

- Active standby pair with a disaster recovery subscriber
- Physical, synchronous Oracle Data Guard
- Oracle RAC

Perform these tasks when large batch jobs must be run on the cached Oracle database tables:

1. Set the autorefresh state to `PAUSED` for cache groups with the `AUTOREFRESH` attribute that are affected by the batch job.
2. Set the autorefresh state to `PAUSED` for cache groups with the `AUTOREFRESH` attribute that are *not* directly affected by the batch job. This ensures that there is a consistent view of the data during batch processing.
3. Run the batch job on the cached Oracle database tables.
4. Make sure all autorefresh change log records have been assigned a valid log sequence number (`logseq`). Call the `cacheInfo.sql` script:

   ```
   % cd timesten_home/install/oraclescripts
   % sqlplus cacheuser/oracle
   SQL> @cacheInfo
   ```

   Look for the number of updates that have not been marked with a valid log sequence number. Ideally, the number should be zero or small (less than 100) for all tables in cache groups for which the autorefresh state was paused. Consider the following example.

   ```
   *************Autorefresh Objects Information  ***************
   Host name: host1
   Timesten datastore name: /scratch/ttuser/ds/mydsn
   Cache table name: TTUSER.NOTAFFECTED
   Change log table name: tt_05_460491_L
   Number of rows in change log table: 1
   Maximum logseq on the change log table: 1
   Timesten has autorefreshed updates up to logseq: 1
   Number of updates waiting to be autorefreshed: 0
   Number of updates that has not been marked with a valid logseq: 0
   ****************************
   Host name: host2
   Timesten datastore name: /scratch/ttuser/ds/mydsn
   Cache table name: TTUSER.AFFECTED
   Change log table name: tt_05_460489_L
   Number of rows in change log table: 100
   Maximum logseq on the change log table: 213
   Timesten has autorefreshed updates up to logseq: 213
   Number of updates waiting to be autorefreshed: 10000
   Number of updates that has not been marked with a valid logseq: 0
   ****************************
   ```

5. For each cache group that was altered in step 1, manually refresh the cache group in parallel mode. Select appropriate values for the transaction size (number of rows committed at a time) and degree of parallelism. For example:

   ```
   REFRESH CACHE GROUP samplecg
    COMMIT EVERY n ROWS PARALLEL m;
   COMMIT;
   ```

   Note that this operation automatically resets the autorefresh state to `ON`.

6. For each cache group that was altered in step 2, set the autorefresh state to `ON`. For example:

```
ALTER CACHE GROUP sampecg2 SET AUTOREFRESH ON;
COMMIT;
```

## Autorefresh not refreshing cache at the specified interval

The following table shows possible causes for autorefresh problems.

| **Possible cause** | **What to do** |
|---|---|
| Cache agent is not started with a cache administration user. | Specify a cache administration user ID and password when starting the cache agent, as shown in "Starting the cache agent" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*. Also see "ttCacheStart" in *Oracle TimesTen In-Memory Database Reference* and "Set the cache administration user name and password" in *Oracle TimesTen Application-Tier Database Cache User's Guide*. |
| Object ID of the base table has changed. | See "Recover and reset autorefresh for the Oracle database objects", below. |
| Autorefresh trigger not enabled. | See "Recover and reset autorefresh for the Oracle database objects", below. |
| Current log sequence number recorded in the `TT_`*`version`*`_USER_COUNT` table is less than to the maximum log sequence number in the autorefresh log table. | See "Recover and reset autorefresh for the Oracle database objects", below. |
| There is no row in the `TT_`*`version`*`_USER_COUNT` table with *`usercount`* > 0 for every active incremental autorefresh table. | See "Recover and reset autorefresh for the Oracle database objects", below. |
| Change log table is empty. | See "Recover and reset autorefresh for the Oracle database objects", below. |
| User count is less than 0 or any `TT_`*`version`*`_USER_COUNT` log sequence anomalies. | See "Recover and reset autorefresh for the Oracle database objects", below. |
| Autorefresh log table, trigger, or sequence associated with a cached table does not exist or is not valid. | Check whether the cache agent was started with the correct cache administration user ID. If the cache administration user ID is correct, follow the procedure described in "Recover and reset autorefresh for the Oracle database objects", below.<br>Check the user error log for messages about "fatal anomalies". This indicates corrupt or missing Oracle database objects. |
| `TT_`*`version`*`_USER_COUNT` table is missing. | Check whether the cache agent was started with the correct cache administration user ID. If the cache administration user ID is correct, follow the procedure in "Recover and reset autorefresh for the Oracle database objects", below.<br>Check the user error log for messages about "fatal anomalies". This indicates corrupt or missing Oracle database objects. |

| Possible cause | What to do |
|---|---|
| Current log sequence number in the `TT_`*`version`*`_USER_COUNT` table changes and is different from the bookmark, and the associated cached table is not refreshed by the next committed autorefresh. | Restart the cache agent. If that does not work, follow the procedure in "Recover and reset autorefresh for the Oracle database objects", below. |
| If autorefresh occurs at the same time you alter the autorefresh mode, state, or interval, you may potentially cause a lock timeout as both actions modify the same table. | Stop the cache agent before altering the autorefresh mode, state, or interval. See "Avoid a lock timeout condition when modifying autorefresh mode, state, or interval" on page 3-19. |
| There is a resource problem. | Restart the cache agent. |

## Reset autorefresh state

Incremental autorefresh does not work if the `TRUNCATE` statement is used on an Oracle database base table. If `TRUNCATE` is used on an Oracle database base table, then you must reset autorefresh by using the `ALTER CACHE GROUP` statement to set the autorefresh state to `OFF` followed by another `ALTER CACHE GROUP` to reset the autorefresh state to `ON`.

## Recover and reset autorefresh for the Oracle database objects

If you know or suspect the Oracle database objects used by autorefresh are the cause of the problem, use the following procedure to recreate the Oracle database objects.

1. Use `ALTER CACHE GROUP` to reset the autorefresh state to `OFF` on all cache groups on all databases that have the affected cached table:

   ```
   ALTER CACHE GROUP cache_group_name SET AUTOREFRESH STATE OFF;
   ```

2. Shut down all cache agents on all affected databases.
3. Check if the user count is zero for each table in the cache group.

   On the Oracle database, execute the following statement:

   ```
   SELECT usercount FROM autorefresh_id.tt_version_user_count
       WHERE tablename ='owner.tablename';
   ```

   If the count is not zero, set the count to zero:

   ```
   UPDATE autorefresh_id.tt_version_user_count SET usercount = 0
       WHERE tablename ='owner.tablename';
   ```

4. Start one of the cache agents. The cache agent performs a clean up operation. It displays the following message to the support log after it has completed the cleanup:

   ```
   Cleanup of the Oracle objects completed
   ```

5. After the cache agent has completed the clean up, use `ALTER CACHE GROUP` to reset the autorefresh state back to `ON`:

   ```
   ALTER CACHE GROUP cache_group_name SET AUTOREFRESH STATE ON;
   ```

6. Start all other cache agents.
7. Use `ALTER CACHE GROUP` to reset the autorefresh state back to `ON` for all of the affected cache groups on all databases.

## Avoid a lock timeout condition when modifying autorefresh mode, state, or interval

A lock timeout may occur if you modify the autorefresh mode, state, or interval with the `ALTER CACHE GROUP` statement at the same time when a scheduled autorefresh starts (scheduled because of the autorefresh interval).

- Autorefresh locks the cache tables and its rows in the `SYS.CACHE_GROUP` table.
- The `ALTER CACHE GROUP SET AUTOREFRESH` statement updates the cache group state in the `SYS.CACHE_GROUP` table.

This results in the following error:

```
Error TT6003: Lock request denied because of time-out.
```

As a workaround, you can:

- Increase the lock timeout with the `ttLockWait` built-in procedure.
- Stop the cache agent before executing the `ALTER CACHE GROUP SET AUTOREFRESH` statement to modify the autorefresh mode, state, or interval. The following example demonstrates setting the autorefresh state to paused.

  1. Stop the cache agent.

     ```
     Command> call ttCacheStop;
     ```

  2. Alter the autorefresh state to the desired state. This example sets the autorefresh state to paused.

     ```
     Command> ALTER CACHE GROUP new_customers SET AUTOREFRESH STATE PAUSED;
     ```

  3. Start the cache agent.

     ```
     Command> call ttCacheStart;
     ```

  For more information on `ALTER CACHE GROUP`, see "ALTER CACHE GROUP" in the *Oracle TimesTen In-Memory Database SQL Reference*. For more information on the `ttLockWait` built-in procedure, see "ttLockWait" in the *Oracle TimesTen In-Memory Database Reference*.

# Incremental autorefresh not progressing

If incremental autorefresh is not progressing, verify that:

- Autorefresh state is `ON`.
- Cache agent is running.

Inspect the support log for the conditions described in the following table:

Table summary is in the first heading cell.

| Condition | What to do |
|---|---|
| Oracle Database server connection errors or warnings | See "Troubleshooting Client/Server problems" on page 2-6 for information about resolving connection problems. |
| Lock timeout errors or warnings on TimesTen | This usually occurs because of an open DDL transaction on the cache group. Commit the DDL transaction so that autorefresh can get the necessary locks. |

| Condition | What to do |
|---|---|
| Insufficient permanent memory region errors on TimesTen | Increase `PermSize`. |
| Autorefresh Oracle Database object validations errors or warnings | See "Recover and reset autorefresh for the Oracle database objects" on page 3-18. |
| Cache agent unexpected exit | Contact TimesTen Customer Support. |
| Core files in main daemon directory | Contact TimesTen Customer Support. |
| Warnings about incremental autorefresh becoming full refresh | See "Incremental autorefresh becomes full autorefresh" on page 3-20. |
| Warnings that autorefresh has not finished for a long time | The autorefresh transaction can take a long time if many transactions have occurred since the last autorefresh.<br>**Note**: Cache groups with the same autorefresh interval are autorefreshed in one transaction. |

## Validate autorefresh for the Oracle database objects

The cache agent automatically verifies that the Oracle database objects exist and that they are valid so that autorefresh can progress. In normal operation, you should not see object validation errors or warnings in the user error log. If you see object validation errors, contact TimesTen Customer Support *unless* one of the following conditions has occurred:

- The TimesTen database has been destroyed without using the `DROP CACHE GROUP` statement.
- A customer application inadvertently modifies the objects directly in the Oracle database.
- A DDL operation occurs on the base table on the Oracle database. This disables the trigger that controls autorefresh operations.

The cache group must be recreated if one of the preceding conditions has occurred.

# Incremental autorefresh becomes full autorefresh

Incremental autorefresh can become full autorefresh if the cache administration user tablespace becomes full.

This section includes the following topics:

- Detecting when incremental autorefresh becomes full
- Understanding the cache administration user tablespace
- Diagnosing a full cache administration user tablespace
- Monitoring the usage of the cache administration user's tablespace
- Considerations when the tablespace of the cache administration is full

## Detecting when incremental autorefresh becomes full

You can detect when incremental autorefresh becomes full refresh by several methods:

- Check for messages in the support log that indicate full autorefresh operations are occurring. For example:

```
2007-08-08 08:06:51.35 Warn: CAC: 11384: TT47166-11384-1087179104-lMarker01403:
```

```
A full autorefresh will be performed for Incremental autorefresh table
USER1.READTAB because change log table TT_06_55555_L on Oracle has been
truncated.
```

- Use the `ttCacheAutorefreshStatsGet` procedure.
  - If autorefresh is *InProgress* for longer than usual, full autorefresh may be occurring.
  - If a much larger number of rows (*`autoRefNumRows`*) was autorefreshed than usual, full autorefresh may have occurred.

  Check the support log for messages about full autorefresh.

## Understanding the cache administration user tablespace

TimesTen strongly recommends creating a separate tablespace for the cache administration user. This tablespace is used as the cache administration user's default tablespace. The tablespace contains autorefresh triggers for each Oracle database table, change log tables for each Oracle database table, and other objects that TimesTen needs for each cache administration user. If you do not specify a separate tablespace, then these objects are placed in the Oracle Database system tablespace.

Specify the tablespace when you create the cache administration user on the Oracle database. You can also specify the tablespace after user creation with the `DEFAULT TABLESPACE` clause of the Oracle Database `ALTER USER` statement.

Change log tables for each of the cached Oracle database tables reside in the cache administration user tablespace. For each update on an Oracle database table, one row (a change log record) is inserted into the change log table for that Oracle database table. The size of a change log record in bytes is as follows:

```
size of change log record = size of primary key on Oracle table + 250
```

The number of records in a change log table depends on the update rate on the Oracle database table and on the autorefresh interval on TimesTen. Every 20 seconds, TimesTen removes change log records that have been applied to all databases that cache the associated Oracle database table.

When change logs are removed, a message similar to the following is displayed in the support log:

```
16:32:26.73 Info: CAC: 5652: TT47112-5652-4756-ogTblGC01036: Garbage collector
deleted 1 rows from TT_06_383270_L where logseq < 1
```

There are options on how to manage what happens when the cache administration user tablespace is filled. See "Considerations when the tablespace of the cache administration is full" on page 3-23 for more information.

## Diagnosing a full cache administration user tablespace

Check for the following conditions if the cache administration user tablespace is full:

- Is the autorefresh state set to `PAUSED`? Change log records accumulate when the state is `PAUSED`.
- Has the cache group been created but not loaded? The default autorefresh state for cache group creation is `PAUSED`.
- Is a cache group being created or is a database being duplicated? Both of these operations temporarily stop clean-up operations on the change log table.

- Are the cache agents on all TimesTen databases running? If a cache agent is not running, change log records accumulate.
- Has a database been abandoned without dropping autorefresh cache groups in the database? Abandoned databases result from scenarios such as the following:
  - The database is destroyed by `ttDestroy -force`.
  - The application connected to the database with the `Overwrite` connection attribute set to 1, but the cache groups that were in the old database are not recreated.

  If the database still exists, connect to the abandoned database and drop the cache group.

Use the `cacheInfo.sql` script to find out how large the change log tables are for each cached Oracle database table. Use the output to verify that the databases are still in use. See "Displaying information from the change log tables" on page 3-12.

If the databases are still in use, verify that the cache agents are running.

Compare the autorefresh progress on TimesTen to the maximum log sequence number on the change log table. If TimesTen is behind, then call the `ttCacheAutorefreshStatsGet` procedure to see whether the autorefresh operations are successful. See "Using the ttCacheAutorefreshStatsGet procedure" on page 3-10.

If the status is *InProgress* longer than seems reasonable, see "Poor autorefresh performance" on page 3-23.

You may need to decrease the autorefresh interval or increase the size of the cache administration user tablespace.

There are options on how to manage what happens when the cache administration user tablespace is filled. See "Considerations when the tablespace of the cache administration is full", following shortly, for more information.

## Monitoring the usage of the cache administration user's tablespace

To monitor the cache administration user tablespace, you can use either Oracle Enterprise Manager alerts or set the TimesTen tablespace threshold parameter.

The cache agent can be configured to periodically monitor the tablespace usage and issue a warning when it exceeds a specified threshold. Set the tablespace threshold percentage with the `TblspaceThreshold` parameter of the `ttCacheConfig` built-in procedure. For example, if you set the `TblspaceThreshold` parameter to 80, then a warning is issued when more than 80% of the tablespace is used.

- If the threshold is set to zero, then no warning is issued. This is the default.
- If the threshold is set between 1 and 99, a warning is issued when the tablespace threshold exceeds that number.
- If the threshold is set to 100, then a warning is issued when the tablespace is full.

For example, to configure for a warning to be issued if the tablespace exceeds 80%, execute the following:

```
call ttCacheConfig('TblspaceThreshold',,,'80');
```

For full details of the `ttCacheConfig` built-in procedure, see the "ttCacheConfig" section in the *Oracle TimesTen In-Memory Database Reference*.

## Considerations when the tablespace of the cache administration is full

With Oracle database tables that are cached in a TimesTen database, you can configure them to use incremental automatic refresh. For these tables, you can specify which one of the following is to occur when the cache administration user's tablespace is full:

- The application performing the DML is to fail. This is the default.

  The tablespace full recovery is set to none. The application receives an "`Out of Tablespace`" error from Oracle Database when the tablespace is full. At that point, the application will need to rollback the transaction.

  Setting the tablespace full recovery to none is configured when you set the `Param` parameter to `TblSpaceFullRecovery` and the `Value` parameter to `None` with the `ttCacheConfig` built-in procedure. For example, the following configures `Param` to `TblSpaceFullRecovery` and `Value` to `None` for the `employees` table that is owned by `terry`:

  ```
  call ttCacheConfig('TblSpaceFullRecovery','terry', 'employees','None');
  ```

- Truncate the change log table to free up space and cause a full autorefresh.

  When the cache administration user's tablespace is full, any application that is executing DML statements on the autorefresh cached Oracle database tables continues to execute. A trigger executes to free up space for new change log records by deleting existing change log records. This can result in a full automatic refresh on cache groups that have the incremental automatic refresh mode configured. However, if the Oracle database table is not configured for incremental automatic refresh, then no trigger executes.

  To set the operation to enable the application to continue and cause an autorefresh, set the `Param` parameter to `TblSpaceFullRecovery` and the `Value` parameter to `Reload` with the `ttCacheConfig` procedure. The user will see stale data until the full autorefresh is complete.

  However, even if the user sets the cache configuration parameter `TblSpaceFullRecovery` with the value of `Reload`, the tablespace may not be able to be emptied enough to handle the case of a growing index. Deleting rows from the change log table may not free up enough space for the index that is on the change log table. If the index is growing so fast that it uses all the tablespace to the point where purging the change log tables does not help, then the user's application may receive the following error:

  ```
  ORA-01654: unable to extend index <index> by 128 in tablespace <tblspace>
  ```

For full details of the `ttCacheConfig` built-in procedure, see the "ttCacheConfig" section in the *Oracle TimesTen In-Memory Database Reference*.

# Poor autorefresh performance

Poor autorefresh performance is usually the result of large autorefresh operations. Use the `ttCacheAutorefreshStatsGet` procedure to check the autorefresh duration and observe whether the status remains *InProgress* for a long time.

Factors that can cause large autorefresh operations include:

- Incremental autorefresh becomes full autorefresh
- Unresponsive or dead TimesTen database degrades autorefresh performance
- Autorefresh cache group refresh with excessive waiting on resources

- Abnormally large change log and base tables degrade autorefresh performance
- Fragmented autorefresh change log table space
- Performance degrades when autorefresh interval is small
- Large autorefresh interval
- Large number of cache groups with the same interval
- High rate of changes to the Oracle database tables
- The number of generations of child tables in a cache group
- The number of rows in the cached Oracle database tables
- The size of the rows in the cached Oracle database tables

Enable an `AUTOREFRESH` trace to diagnose autorefresh performance problems. See "AUTOREFRESH tracing" on page 1-14.

## Unresponsive or dead TimesTen database degrades autorefresh performance

**Note:** Automatic recovery for TimesTen cache groups only applies to read-only and user managed cache groups that use the `AUTOREFRESH` cache group attribute. In this section, all references to autorefresh cache groups are read-only and user managed cache groups that use the `AUTOREFRESH` cache group attribute.

If any TimesTen databases containing autorefresh cache groups are destroyed or no longer in use, TimesTen continues to track autorefresh changes to the Oracle database tables for the TimesTen database for which the cache agent is not running. This causes automatic refresh to cache groups in active TimesTen databases to slow down.

The cache agent is responsible for detecting if a database is unresponsive or no longer in use. You can specify if and how a dead TimesTen database is to be recovered. However, you cannot recover a TimesTen database if all of the Oracle database objects have been removed.

The following sections describe how you can avoid a degraded autorefresh performance for inactive TimesTen databases:

- Setting cached TimesTen database timeout
- Configuring recovery method for certain cache groups

### Setting cached TimesTen database timeout

You can instruct TimesTen to mark the database as dead and no longer accepting updates if the cache agent has not communicated with the Oracle Database server within a specific timeout period.

Set the timeout for the TimesTen database and the recovery method for each autorefresh cache group with the `AgentTimeOut` parameter in the `ttCacheConfig` built-in procedure. The timeout value applies to the all TimesTen databases that use the same cache administration user. You should set the timeout value greater than the time necessary to load the TimesTen database into memory on first connect and start the cache agent. Otherwise, the TimesTen database could be incorrectly marked as dead. For any planned maintenance for the TimesTen instance, you could temporarily set the `AgentTimeOut` value to zero to disable the timeout. For full details of the `ttCacheConfig` built-in procedure, see the "ttCacheConfig" section in the *Oracle TimesTen In-Memory Database Reference*.

For example, the following sets the timeout value for the TimesTen database to 6000 seconds or 100 minutes. If the cache agent does not contact the Oracle Database server within a 100-minute period, then the TimesTen database is marked as dead.

```
ttIsql> call ttCacheConfig('AgentTimeOut',,,'6000');
```

### Configuring recovery method for certain cache groups

You can recover a TimesTen database and autorefresh cache groups if they are not synchronizing with the Oracle database. If there is no synchronization, then updates on the Oracle database tables are not automatically refreshed to the corresponding TimesTen cache tables.

You can configure the `DeadDbRecovery` parameter of the `ttCacheConfig` built-in procedure to specify how to recover the synchronization for the TimesTen database and all autorefresh cache groups. The setting for `DeadDbRecovery` applies to all TimesTen databases that use the same cache administrator user. Set the `DeadDbRecovery` parameter to `Normal`, `Manual` or `None` to describe how TimesTen is to recover the database and all autorefresh cache groups. The `DeadDbRecovery` setting applies to all TimesTen databases that use the same cache administration user. While TimesTen is recovering the database and its autorefresh cache groups, there is an autorefresh status for the TimesTen database and the autorefresh cache groups that describes the recovery status for each of these entities. The TimesTen database can have an automatic refresh status of Alive, Dead or Recovering. The autorefresh cache groups can have an automatic refresh status of OK, Dead or Recovering. The TimesTen database status changes are linked to changes in the status for the autorefresh cache groups, as follows:

- If the recovery method is set to Normal, then when TimesTen starts a full automatic refresh on an autorefresh cache group, the cache group's status is set to Recovering and the database's status is also set to Recovering.
- The TimesTen database's status is only set to Alive when all of the autorefresh cache groups have either been recovered to OK or have been dropped.
- When the database status is set to Dead, then all of its autorefresh cache groups are also set to Dead.

**Note:** You can determine the autorefresh status of the TimesTen database and autorefresh cache groups with the `ttCacheDbCgStatus` built-in procedure, which is described in the "ttCacheDbCgStatus" section in the *Oracle TimesTen In-Memory Database Reference*.

When communication between the cache agent and the Oracle Database server is reestablished, TimesTen determines how to recover the autorefresh cache groups. TimesTen follows the recovery method you configured in the `DeadDbRecovery` parameter in the `ttCacheConfig` built-in procedure. This parameter can be set to one of the following:

- `Normal`: This is the default. The autorefresh cache groups will each be recovered with a full automatic refresh. After the first full refresh, the cache group is recovered and will incrementally perform autorefresh.

The autorefresh cache groups within the same automatic refresh interval will be transactionally consistent. Because it is a full refresh, it is not as performant as an incremental refresh.

The autorefresh sets the status to Recovering. When the full autorefresh is completed successfully, the autorefresh cache group status is set to OK.

- `Manual`: You must manually refresh an autorefresh cache group to recover it, or unload it if the cache group is dynamic.
- `None`: The autorefresh cache group will never be recovered by a TimesTen autorefresh. Drop and recreate the cache group to recover it.

The database status changes as the first autorefresh cache group status changes. If there is at least one cache group that is in the process of recovery, then the database status is set to Recovering. Once all cache groups have been recovered, the status of the TimesTen database is marked as Alive.

The following example sets the `DeadDbRecovery` parameter to `Normal` for all autorefresh cache groups. The dead TimesTen database will be recovered when all of its autorefresh cache groups have each been recovered with a full automatic refresh.

```
ttIsql> call ttCacheConfig('DeadDbRecovery',,,'Normal');
```

When TimesTen databases participating in an active standby pair replication scheme contains cache groups, if the autorefresh status of the active master database is Dead and the autorefresh status of the standby master database is Alive, the standby master does not automatically assume the role of the active master. The recovery requires that you manually ensure that the cache and replication agents are executing. The specifics for each situation is as follows:

*Table 3–2 Recovery for cache groups involved in active standby replication pair*

| DeadDbRecovery Setting | Active Master | Standby Master | Resulting Behavior |
|---|---|---|---|
| Normal | Alive | Dead | Make sure that the cache and replication agents are executing on the standby master. Once the cache agent can connect to the Oracle Database, then the status of all autorefresh cache groups is set to Recovering. This sets the database to Recovering. Once a single cache group has received enough data to resume autorefresh, the status is set to OK. After all cache group are set to OK, the database is set to Alive.<br>Alternatively, you can fail the standby master and rollout a new standby master. |
| Normal | Dead | Alive | Make sure that the cache and replication agents are executing on the active master. Once the cache agent can connect to the Oracle Database, then the status of all autorefresh cache groups is set to Recovering. This sets the database to Recovering. Once a single cache group has received enough data to resume autorefresh, the status is set to OK. After all cache group are set to OK, the database is set to Alive.<br>Alternatively, you can fail the active master, switch the standby master as the new active and then rollout a new standby master. |
| Normal | Dead | Dead | Make sure that the cache and replication agents are executing on both masters. Once the cache agent can connect to the Oracle Database, then the status of all autorefresh cache groups is set to Recovering. This sets the database to Recovering. Once a single cache group has received enough data to resume autorefresh, the status is set to OK. After all cache group are set to OK, the database is set to Alive.<br>Alternatively, you can rollout new masters. |
| Manual | Alive | Dead | Make sure that the cache and replication agents are executing on the standby master. Once the cache agent can connect to the Oracle Database, then the status of all autorefresh cache groups is set to Recovering. This sets the database to Recovering. Once a single cache group has received enough data to resume autorefresh, the status is set to OK. After all cache group are set to OK, the database is set to Alive.<br>Alternatively, you can fail the standby master and rollout a new standby master. |
| Manual | Dead | Alive | Make sure that the cache and replication agents are executing on the active master. Use a manual refresh to recover the autorefresh cache groups on the active master. After all cache group are set to OK or have been dropped, the database is set to Alive. |
| Manual | Dead | Dead | Make sure that the cache and replication agents are executing on the active master. Use a manual refresh to recover the autorefresh cache groups on the active master. After all cache group are set to OK or have been dropped, the database is set to Alive. Changes are then replicated to the standby master. |
| None | Alive | Dead | Mark the standby master as failed. Execute `ttDestroy` utility for the standby master database. Duplicate the active master by executing `ttRepAdmin -duplicate` utility from the active master. |
| None | Dead | Alive | Destroy the dead active master with the `ttDestroy` utility. Recover the dead active master by duplicating the standby master with the `ttRepAdmin -duplicate` utility. |
| None | Dead | Dead | Roll out new masters. |

## Autorefresh cache group refresh with excessive waiting on resources

During an autorefresh cache group refresh, there can be excessive buffer busy waits, row lock waits, and deadlocks on updates in the Oracle database, which can negatively affect the throughput performance. When there are multiple deadlocks on updates in the Oracle database involving the autorefresh log tables, the following may appear in the support log:

```
Oracle native error code = 60, msg = ORA-00060: deadlock detected while waiting
for resource
An error occurred while preparing or executing the following Oracle sql
statement: <some statement involving <cache admin user>.TT_##_#######_L  where
the # is some number>
```

You can improve your performance by modifying the `INITRANS` and `FREELISTS` settings, which can affect the concurrent inserts into the autorefresh log table and internal maintenance of these tables. The application updating the base table that is being autorefreshed encounters a throughput performance hit when these settings are not appropriately configured.

You can automatically or manually manage these settings as follows:

- Use ASSM tablespace, which automatically manages `FREELISTS`.
- Manually adjust `FREELISTS` and `INITRANS` for the autorefresh log table on the Oracle database.

The following details how to manually modify `INITRANS` and `FREELISTS` for the autorefresh log table on the Oracle database:

1. Retrieve the name of the autorefresh log table that is on the Oracle database.

   Under the cache administration user login, execute the SQL*Plus script `cacheInfo.sql` that lists the autorefresh change log table name, along with other items. The following example executes the `cacheInfo.sql` script that lists the autorefresh change log table name as `tt_06_1216726_L`, as shown in bold:

```
SQL> @cacheInfo.sql
*************Autorefresh Objects Information  ***************
Host name: syst
Timesten datastore name: /users/OracleCache/alone1
Cache table name: ORATT.ORDERS
Change log table name: tt_06_1216726_L
Number of rows in change log table: 1
Maximum logseq on the change log table: 2
Timesten has autorefreshed updates upto logseq: 1
Number of updates waiting to be autorefreshed: 1
Number of updates that has not been marked with a valid logseq: 0
****************************
Host name: consyst
Timesten datastore name: /users/OracleCache/alone1
Cache table name: ORATT.ITEMS
Change log table name: tt_06_1279699_L
Number of rows in change log table: 7
Maximum logseq on the change log table: 0
Timesten has autorefreshed updates upto logseq: 0
Number of updates waiting to be autorefreshed: 5
Number of updates that has not been marked with a valid logseq: 5
****************************
```

2. Manually alter the table on the Oracle database. The following example uses the table from the previous example. This example alters the `INITRANS` and `FREELISTS` settings for the `bar.tt_06_1279699_L` table.

```
ALTER TABLE BAR.TT_06_1279699_L INITRANS 10;
ALTER TABLE BAR.TT_06_1279699_L STORAGE(FREELISTS 5);
or
ALTER TABLE BAR.TT_06_1279699_L MOVE STORAGE(FREELISTS 5);
```

3. Alter the `INITRANS` and `FREELISTS` settings for the index for this table, which have the same name as the autorefresh change log table with an additional "L" at the end of it. For example, the index for table `bar.tt_06_1279699_L` is `bar.tt_06_1279699_LL`.

   These settings should be the same as what you set for the autorefresh change log table.

```
ALTER INDEX BAR.TT_06_1279699_LL INITRANS 10;
ALTER INDEX BAR.TT_06_1279699_LL STORAGE(FREELISTS 5);
```

## Abnormally large change log and base tables degrade autorefresh performance

The cache thread SQL refresh joins the change log table and the base table, which identifies rows needed to be refreshed into TimesTen. The larger the cardinalities of the base table and the change log table, the longer the time necessary to perform this join. Performance degradation may occur if either the change log table or the base table is abnormally large.

The following describes scenarios where the change log table can become abnormally large.

- If the change log table is never purged in configurations where cache groups from multiple DSNs all reference the same base table, it increases in size indefinitely. If one or more of the cache agents for these groups are turned off, those DSNs will not properly refresh their cache groups and the change log tables will not be purged. If the autorefresh state is turned to paused on one of multiple nodes, the other nodes may slow down.
- The change log table can grow abnormally large if some cache agents have been shut down. Resolve this issue by restarting the cache, which will purge all of the backlogged log rows to be purged and all of the cache groups to be synchronized after the completion of the refresh cycle for all cache groups.
- The change log table can be abnormally large if rows inserted into the change log table are never purged and can never be purged by normal processing. This occurs when one or more DSNs are destroyed or rebuilt without first removing the cache groups. The cache group tables on the Oracle database have no information that the cache groups have been destroyed, which corrupts the entire cache group. Rebuild and reinitialize all of the cache groups associated with this base table. Alternatively, never destroy a DSN with cache groups. Instead, always drop the cache groups before destroying a DSN.

## Fragmented autorefresh change log table space

Change log tables can become fragmented when a high water mark occurs as the result of change logs building up when TimesTen is shut down, for example. If change log tables have become fragmented, you can:

- Coalesce their indexes. This can be done without preventing DML changes to the base tables.

- Perform an online segment shrink. This can be done without preventing DML changes to the base tables.
- Rebuild the change log tables.

Check to see whether space is being wasted:

1. Determine the name of the change log table by running the `cacheInfo.sql` script on the Oracle database.
2. Calculate the size of the change log table. Call the result A. Adapt the name of the change log table in this example.

   ```
   SELECT table_name, ROUND((BLOCKS*8),2)||'KB' "size"
    FROM user_tables
    WHERE table_name LIKE 'TT_05_%_L";
   ```

3. Calculate the size of the data in the change log table. Call the result B. Adapt the name of the change log table in this example.

   ```
   SELECT table_name, ROUND((num_rows*avg_row_length/1024),2)|| 'KB' "size"
    FROM user_tables
    WHERE table_name LIKE 'TT_05_%_L';
   ```

4. If `(B/A)*100` is greater than 50 percent, then there is at least 40 percent space wasted (assuming a `PCTFREE` storage parameter set to 10). If there is at least 40 percent space wasted, defragmenting the change log table is recommended.

Perform these steps to defragment the change log table:

1. Alter the cache group to set the autorefresh state to `PAUSED`.
2. Copy the rows in the change log table to a temporary table.
3. Truncate the change log table.
4. Insert the rows from the temporary table to the change log table.
5. Alter the cache group to set the autorefresh state to `ON`.

### Performance degrades when autorefresh interval is small

When a relatively short refresh interval, such as a few hundred milliseconds, is combined with a large number of entries in the log table or in the base table, a cache refresh operation does not complete before the next refresh operation is scheduled to begin. In this case, the entries in the log table can be un-marked when the current autorefresh cycle finishes.

Thus, the same rows can be refreshed from the base table to the cache group in the next autorefresh cycle, by which time the rows will be marked. Make sure that the time it for the refresh is greater than the refresh interval. Set the refresh interval to a value where redundant refreshes will not occur.

## Declaring NOVALIDATE on constraints causes cache group creation failure

If the Oracle database table on which you want to create the cache group declares `NOVALIDATE` on columns with primary key, `UNIQUE` or `NOT NULL` constraints, the creation of the cache group fails.

---

**Note:** This does not apply to any foreign key constraints. However, TimesTen recommends that any matching foreign key is in the enabled `VALIDATE` state. Your workload performance may be affected when you alter a foreign key column to the enabled `VALIDATE` state.

---

TimesTen perceives a `NOVALIDATE` on a primary key or `NOT NULL` table column definition as a `NULL` and, therefore, not qualified as a column on which to build the cache group. Thus, all columns with the primary key, `UNIQUE` and `NOT NULL` column constraints must be enabled with the `VALIDATE` state when creating a cache group from the Oracle database table.

When you create a cache group from an Oracle database table with one or more of these constraints, the following errors are thrown:

```
5124: Autorefresh/propagate are not allowed on restricted cache group
5168: Restricted cache groups are deprecated
5120: No matching unique index with not null columns, unique key constraint
 with not null columns, or primary key constraint on table EVENTLOG, cache
 operations are restricted.
```

If you receive these errors, you can perform a `SELECT` statement to verify any existing `NOVALIDATE` constraints on the Oracle database table. The following `SELECT` statement shows all constraints on the `MyTable` table:

```
SQL> select constraint_name, constraint_type, validated, status from
        all_constraints where table_name = 'MyTable';

CONSTRAINT_NAME                C VALIDATED     STATUS
------------------------------ - ------------- --------
REFID_CONSTRAINT               C VALIDATED     ENABLED
PKEY_CONSTRAINT                P NOT VALIDATED DISABLED
```

If the table column that is to be the primary key for the cache table is enabled as `NOVALIDATE`, perform the following steps to enable the column with the `VALIDATE` state:

1. Enable the `NOVALIDATE` state for the primary key column.

   ```
   SQL> alter table MyTable modify constraint PKEY_CONSTRAINT
            enable novalidate;
   Table altered.

   SQL> select constraint_name, constraint_type, validated, status
             from all_constraints where table_name = 'MyTable';

   CONSTRAINT_NAME                C VALIDATED     STATUS
   ------------------------------ - ------------- --------
   REFID_CONSTRAINT               C VALIDATED     ENABLED
   PKEY_CONSTRAINT                P NOT VALIDATED ENABLED
   ```

2. Enable the `VALIDATE` state for the primary key column.

   ```
   SQL> alter table MyTable modify constraint PKEY_CONSTRAINT validate;
   Table altered.

   SQL> select constraint_name, constraint_type, validated, status
             from all_constraints where table_name = 'MyTable';

   CONSTRAINT_NAME                C VALIDATED     STATUS
   ------------------------------ - ------------- --------
   REFID_CONSTRAINT               C VALIDATED     ENABLED
   ```

```
    PKEY_CONSTRAINT                P VALIDATED     ENABLED
```

## AWR report showing lock contention with DBMS_LOCK

There may be some concern about lock contention when seeing `DBMS_LOCK` in the Automated Workload Repository (AWR) Report. This wait event is the garbage collector session trying to place a hold on a resource that another garbage collector session from another database has already locked. Thus, only the current garbage collector session waits. The wait for the garbage collector process does not block other processes, except other garbage collectors.

For example, the following shows a contention event in the AWR report:

```
AWR

Top 5 Timed Events                                        Avg %Total
~~~~~~~~~~~~~~~~~~                                       wait   Call
Event                                 Waits    Time (s)   (ms)   Time Wait Class
------------------------------ ------------ ----------- ------ ------ ----------
enq: UL - contention                  2,388       6,997   2930   72.0 Application
```

In addition, only a small amount of CPU time is used for the garbage collector, as shown in the "SQL ordered by CPU Time" section in the PERF AWR report.

```
SQL ordered by CPU Time             DB/Inst: REM0LNX/REM  Snaps: 14976-14977
-> Resources reported for PL/SQL code includes the resources used by all SQL
   statements called by the code.
-> % Total DB Time is the Elapsed Time of the SQL statement divided
   into the Total Database Time multiplied by 100

    CPU      Elapsed                  CPU per  % Total
  Time (s)   Time (s)  Executions     Exec (s) DB Time    SQL Id
---------- ---------- ------------ ----------- ------- -------------
         0      3,508          120        0.00    36.1 0mt5pk2501gph
Module: timestenorad@etcpro01.oracle.com (TNS V1-V3)
DECLARE v_lockHandle VARCHAR2(200); BEGIN dbms_lock.allocate_unique(
'ORATT$ORA_GC1_CACHEADMIN', v_lockHandle); :retval := dbms_lock.request(
v_lockHandle, dbms_lock.x_mode, 30, FALSE); END;
         0      3,499          120        0.00    36.0 bb07h2a1v817x
Module: timestenorad@etcpro01.oracle.com (TNS V1-V3)
DECLARE v_lockHandle VARCHAR2(200); BEGIN dbms_lock.allocate_unique(
'ORATT$ORA_DDSMONITOR1_CACHEADMIN', v_lockHandle); :retval :=
dbms_lock.request(v_lockHandle, dbms_lock.x_mode, 30, FALSE); END;
```

# 4

# Troubleshooting Replication

The following sections in this chapter describe how to troubleshoot problems you may encounter when replicating databases:

- Unable to create a replication scheme
- Unable to alter a replication scheme
- Unable to start or stop replication agent
- Replication does not work
- Replication unresponsive, appears hung
- Poor replication or XLA performance
- Problems using ttRepAdmin
- Problems with conflict checking
- Problems with parallel replication

> **Note:** Error log messages from the replication agent daemon are designated by `REP` in the message, as shown in several examples in this chapter. These messages are enabled unless you disable the `REPLICATION` component of the `ttDaemonLog` utility. See "ttDaemonLog" in *Oracle TimesTen In-Memory Database Reference* for information about `ttDaemonLog` options.

## Unable to create a replication scheme

This section describes what to check if you are unable to use `CREATE REPLICATION` to create a replication scheme.

| Possible cause | What to do |
|---|---|
| User does not have `ADMIN` privilege. | You must have `ADMIN` privilege to use the `CREATE REPLICATION` or `DROP REPLICATION` statements. |
| Database name, host name, or element name is incorrect. | ■ Check the `CREATE REPLICATION` statement for typographical errors.<br>■ See "Check host names" on page 4-7.<br>■ Use official host names instead of aliases.<br>■ The host name should match the value returned by the `hostname` command on your system and should be used consistently throughout the replication scheme. |

| Possible cause | What to do |
|---|---|
| The local host is not part of the replication scheme. | Create the replication scheme on a host that will be part of the replication scheme. |
| Replication tables defined in the `CREATE REPLICATION` statement do not exist. | The name, owner, and column definitions of the tables participating in the replication scheme must be identical on both the master and subscriber databases. Use `CREATE TABLE` to create tables on the database, or use the `ttRepAdmin -duplicate` utility or the `ttRepDuplicateEx` C function to duplicate the entire database to be replicated. |
| There are other problems. | Review the procedures and requirements described in "Defining Classic Replication Schemes" in the *Oracle TimesTen In-Memory Database Replication Guide*. |

## Unable to alter a replication scheme

This section describes what to check if you are unable to use `ALTER REPLICATION` to alter a replication scheme.

| Possible cause | What to do |
|---|---|
| User does not have `ADMIN` privilege. | You must have `ADMIN` privilege to use the `ALTER REPLICATION` statement. |
| Replication agent is in Start state. | Most `ALTER REPLICATION` operations are supported only when the replication agent is stopped (`ttAdmin -repStop`). Stop the replication agents on both master and subscriber databases, alter the replication scheme on both master and subscriber databases, then restart both replication agents. |
| Database name, host name, or element name is inconsistent. | ■ Check `ALTER REPLICATION` statement for typographical errors.<br>■ See "Check host names" on page 4-7. |
| Replication table defined in the `ALTER REPLICATION` statement does not exist. | Use `CREATE TABLE` to create a table on the database. |
| There are other problems. | Review the procedures and requirements described in "Altering a Classic Replication Scheme" in the *Oracle TimesTen In-Memory Database Replication Guide*. |

## Unable to start or stop replication agent

This section describes what to check if you are unable to start or stop a replication agent.

| Possible cause | What to do |
|---|---|
| User does not have `ADMIN` privileges. | You must have `ADMIN` privileges to use the `ttAdmin` utility or the `ttRepStart` or `ttRepStop` procedures to start or stop a replication agent. |
| TimesTen daemon is not started. | Check the state of the TimesTen daemon, as described in "Check the TimesTen user error log" on page 2-2. If necessary, start the TimesTen daemon as described in "Working with the TimesTen Daemon" in the *Oracle TimesTen In-Memory Database Operations Guide*. |

| Possible cause | What to do |
|---|---|
| Database does not participate in a replication scheme. | If a database does not participate in a replication scheme, attempts to start a replication agent for that database will fail. Use `CREATE REPLICATION` to create a replication scheme for the database. |

## Replication does not work

If you are unable to get replication working between a master and subscriber database, the problem may be one or more of the following:

| Possible cause | See... |
|---|---|
| TimesTen daemon or replication agents are not running. | "Check status of TimesTen daemon and replication agents", below |
| Master and subscriber agents are not communicating. | "Check that replication agents are communicating" on page 4-5 |
| Replication is not in Start state. | "Check replication state" on page 4-5 |
| There is an error in the replication scheme. | "Check replication scheme configuration" on page 4-5 |
| Owner names for replication scheme and tables are inconsistent. | "Check owner names" on page 4-7 |
| Replication tables are inconsistent. | "Check consistency between replicated tables" on page 4-8 |

### Check status of TimesTen daemon and replication agents

Use the `ttStatus` utility to confirm the main TimesTen daemon is running and the replication agents are started for all of your master and subscriber databases. The output from a simple replication scheme using a single master and subscriber database should look like that shown in Example 4–1.

If the TimesTen daemon is running, but the replication agents are not, the output looks like that shown in Example 4–2. In this case, start the replication agents as described in "Starting and stopping the replication agents" in the *Oracle TimesTen In-Memory Database Replication Guide*.

If neither the TimesTen daemon or replication agents are running, the output looks like that shown in Example 4–3. In this case, confirm you have correctly installed TimesTen and the Data Manager service is started.

***Example 4–1 ttStatus output for one master and one subscriber***

```
C:\>ttStatus
TimesTen status report as of Mon Aug  6 22:07:53 2012
Daemon pid 5088 port 17000 instance MYINSTANCE
TimesTen server pid 4344 started on port 17002
------------------------------------------------------------------------
Data store c:\temp\subscriber1ds
There are 12 connections to the data store
Data store is in shared mode
Shared Memory KEY Global\DBI45b9471c.2.SHM.2 HANDLE 0x280
Type            PID     Context     Connection Name              ConnID
Process         1244    0x00d08fb0  subscriber1ds                     1
Replication     4548    0x00aed2f8  LOGFORCE                          4
Replication     4548    0x00b03470  TRANSMITTER                       5
Replication     4548    0x00b725a8  RECEIVER                         6
Replication     4548    0x00b82808  REPHOLD                          2
Replication     4548    0x00b98980  REPLISTENER                      3
Subdaemon       2752    0x00526768  Worker                        2042
Subdaemon       2752    0x0072a758  Flusher                       2043
Subdaemon       2752    0x007308c0  Checkpoint                    2044
Subdaemon       2752    0x00736a28  HistGC                        2046
Subdaemon       2752    0x067f02f8  Aging                         2045
Subdaemon       2752    0x068364a0  Monitor                       2047
Replication policy  : Manual
Replication agent is running.
Cache agent policy : Manual
------------------------------------------------------------------------
Data store c:\temp\masterds
There are 12 connections to the data store
Data store is in shared mode
Shared Memory KEY Global\DBI45b945d0.0.SHM.6 HANDLE 0x2bc
Type            PID     Context     Connection Name             ConnID
Process         5880    0x00d09008  masterds                         1
Replication     3728    0x00aed570  LOGFORCE                         4
Replication     3728    0x00b036e8  TRANSMITTER                      5
Replication     3728    0x00b168b8  REPHOLD                          3
Replication     3728    0x00b1ca20  REPLISTENER                      2
Replication     3728    0x00b22b88  RECEIVER                         6
Subdaemon       3220    0x00526768  Worker                        2042
Subdaemon       3220    0x0072e768  Flusher                       2043
Subdaemon       3220    0x007348d0  Checkpoint                    2044
Subdaemon       3220    0x067b0068  Aging                         2045
Subdaemon       3220    0x067c0040  Monitor                       2047
Subdaemon       3220    0x068404c8  HistGC                        2046
Replication policy  : Manual
Replication agent is running.
Cache agent policy : Manual
------------------------------------------------------------------------
Data store c:\temp\demo
There are no connections to the data store
Replication policy : Manual
Cache agent policy : Manual
------------------------------------------------------------------------
End of report
```

***Example 4–2    Replication agent is not running***

```
> ttStatus
TimesTen status report as of Mon Aug  6 22:07:53 2012

Daemon pid 3396 port 15000 instance MYINSTANCE
TimesTen server pid 3436 started on port 15002
-----------------------------------------------------------------
Data store c:\temp\subscriberds
There are no connections to the data store
cache agent restart policy: manual
-----------------------------------------------------------------
Data store c:\temp\masterds
There are no connections to the data store
cache agent restart policy: manual
-----------------------------------------------------------------
End of report
```

***Example 4–3 TimesTen daemon and replication agent are not running***

```
> ttStatus
ttStatus: Could not connect to TimesTen daemon: Connection refused
```

## Check that replication agents are communicating

Use `ttRepAdmin -receiver -list` to see that the replication agents are communicating with each other. If the `masterds` database is replicating to `subscriberds`, the output should look similar to the following:

***Example 4–4 Check that the replication agents are communicating***

```
> ttRepAdmin -receiver -list masterDSN
Peer name         Host name                 Port    State  Proto
----------------  ------------------------ ------  ------- -----
SUBSCRIBERDS      MYHOST                    Auto   Start       10

Last Msg Sent Last Msg Recv Latency TPS     RecordsPS Logs
------------- ------------- ------- ------- --------- ----
0:01:12       -               19.41       5        52    2
```

## Check replication state

Use the `ttReplicationStatus` procedure to check the state of the subscriber database with respect to its master. If the subscriber is in the `Stop`, `Pause`, or `Failed` state, use the `ttReplicationStatus` procedure to reset the subscriber state to `Start`, as described in "Set the replication state of subscribers" in the *Oracle TimesTen In-Memory Database Replication Guide*.

***Example 4–5 Obtain status of the subscriber database from the master database***

Use `ttReplicationStatus` to obtain the status of the `subscriberds` database from its master database, `masterDSN`, enter:

```
> ttIsql masterDSN
Command> CALL ttReplicationStatus ('subscriberds');
< SUBSCRIBERDS, MYHOST, 0, pause, 1, 10, REPSCHEME, REPL >
1 row found.
```

To reset state to Start call the `ttRepSubscriberStateSet` procedure:

```
Command> CALL ttRepSubscriberStateSet('REPSCHEME', 'REPL', 'SUBSCRIBERDS',
'MYHOST', 0)
Command> CALL ttReplicationStatus ('subscriberds');
< SUBSCRIBERDS, MYHOST, 0, start, 1, 152959, REPSCHEME, REPL >
1 row found.
```

## Check replication scheme configuration

This section describes some procedures you can use to confirm the correct configuration of the various components in your replicated system. The basic procedure categories are:

- Check ttRepAdmin -showconfig
- Check the TTREP.TTSTORES table
- Check host names

## Check ttRepAdmin -showconfig

Use `ttRepAdmin -showconfig` to confirm the configuration of your replication scheme.

What to look for:

- Are all of the subscriber agents started and reported to be in the Start state? If not, reset the agents to the Start state. See "Set the replication state of subscribers" in the *Oracle TimesTen In-Memory Database Replication Guide*.
- Do the reported Peer names match the names given in the `DataStore` attributes in the DSN definitions for the replicated databases? Replication does not work if you specified the names given for the Data Source `Name` attributes.
- Is there anything under List of subscribers? If not, confirm the database names you specified in the DSN definition are consistent with those you specified in your replication scheme configuration file.
- Are the Host names correct? If in doubt, see "Check host names" on page 4-7.
- Are the correct table names displayed under Table details? If not, correct the table names in your replication scheme configuration file.

***Example 4–6 Confirm the configuration of the replication scheme***

```
> ttRepAdmin -showconfig masterDSN
Self host "MYHOST", port auto, name "MASTERDS", LSN 4/2970276, timeout 120,
threshold 0
List of subscribers
-----------------
Peer name         Host name                Port    State  Proto
----------------  ------------------------ ------  ------- -----
SUBSCRIBERDS      MYHOST                   Auto    Start     10
Last Msg Sent Last Msg Recv Latency TPS     RecordsPS Logs
------------- ------------- ------- ------- --------- ----
0:01:12       -               19.41       5        52    2
List of tables and subscriptions
--------------------------------
Table details
-------------
Table : REPL.TAB
Master Name               Subscriber Name
-----------               -------------
MASTERDS                  SUBSCRIBERDS
```

## Check the TTREP.TTSTORES table

Check the `TTREP.TTSTORES` table to confirm that replication associates the replication scheme with the local database.

***Example 4–7 Confirm that the replication scheme is associated with the local database***

Connect to the database and enter:

```
SELECT * FROM ttrep.ttstores WHERE is_local_store <> 0x0;
Command> select * from ttrep.ttstores where is_local_store <> 0x0;
< -5193371075573733683, MYHOST, MASTERDS, 01, 0, 0, 4, 0 >
1 row found.
```

There should be exactly one row returned. If more than one row is returned, contact TimesTen Customer Support. If no rows are returned, then none of the hosts returned by the following statement is perceived to be a local system by TimesTen replication:

```
SELECT DISTINCT host_name FROM ttrep.ttstores;
```

It may also be that none of the database names specified in your replication scheme match those specified in your DSN descriptions.

### Check host names

Some hosts or IP addresses specified in a replication scheme cannot be resolved by the replication agent because:

- Host names or IP addresses specified in the replication scheme are wrong or misspelled.
- Host names or IP addresses cannot be resolved or found by DNS or in the `/etc/hosts` file
- Entries in the `/etc/hosts` file are incorrectly ordered in appearance. This error is most common when multiple NICs are used. You must have root privilege to make changes to the `/etc/hosts` files.

See "Configuring the network" in the *Oracle TimesTen In-Memory Database Replication Guide* for details on how to configure DNS and `/etc/hosts` files for host systems used for replication.

To check if a host name in the replication scheme matches the host name of the local system, write an application to perform these tasks:

1. Use a `gethostname` operating system function call to determine the host name of the running host.
2. Call `gethostbyname` with the output from Step 1.
3. Call `gethostbyname` with the host name specified in the replication scheme.
4. Compare output of Step 2 and Step 3. If there is a match, then the running host is involved in replication. Otherwise, it is not involved in replication.

## Check owner names

As described in "Table requirements and restrictions for classic replication schemes" and "Owner of the classic replication scheme and replicated objects" in the *Oracle TimesTen In-Memory Database Replication Guide*, the owner names of your replication scheme and your replicated tables must be consistent across all participating databases. This indicates that the replication scheme owner must match in all databases and each table must be owned by the same user in each database.

### Checking replication owner

Check the owner name assigned to your replication scheme by calling the `ttIsql repschemes` command or by listing the contents of the `TTREP.REPLICATIONS` table.

Example 4–8 shows that the replication scheme name, `REPSCHEME`, has a consistent owner name (`REPL`) in the databases on both `SYSTEM1` and `SYSTEM2`. Example 4–9 shows the scheme name with inconsistent owner names. This can occur if you omit the owner name from the replication scheme definition and the system uses the Id of the replication scheme creator.

***Example 4–8 Consistent owner names for replication scheme***

On `SYSTEM1`:

```
> ttIsql masterDSN
```

```
Command> select * from ttrep.replications;
< REPSCHEME           , REPL                 , C, 0, 0, -1 >
1 row found.
```

On `SYSTEM2`:

```
> ttIsql -connStr "dsn=subscriberDSN"
Command> select * from ttrep.replications;
< REPSCHEME           , REPL                 , C, 0, 0, -1 >
1 row found.
```

***Example 4–9 Inconsistent owner names for replication scheme***

On `SYSTEM1`:

```
> ttIsql masterDSN
Command> select * from ttrep.replications;
< REPSCHEME           , SYSTEM1              , C, 0, 0, -1 >
1 row found.
```

On `SYSTEM2`:

```
> ttIsql -connStr "dsn=subscriberDSN"
Command> select * from ttrep.replications;
< REPSCHEME           , SYSTEM2              , C, 0, 0, -1 >
1 row found.
```

### Checking table owner

Check the owner names assigned to the tables in each database by using the `ttIsql tables` command.

***Example 4–10 Consistent table owner names***

This example shows that the `TAB` table has a consistent owner name (`REPL`) in the databases on both `SYSTEM1` and `SYSTEM2`.

| Output for SYSTEM1 | Output for SYSTEM2 |
|---|---|
| REPL.TAB | REPL.TAB |

***Example 4–11 Inconsistent table owner names***

This example shows the `TAB` table with inconsistent owner names, which were automatically assigned for each host.

| Output for SYSTEM1 | Output for SYSTEM2 |
|---|---|
| SYSTEM1.TAB | SYSTEM2.TAB |

## Check consistency between replicated tables

By default, TimesTen creates a replication scheme with the `TABLE DEFINITION CHECKING RELAXED` attribute. The `TABLE DEFINITION CHECKING RELAXED` attribute does not require that the column definition of replicated tables be identical. Rather, the `RELAXED` attribute only requires that replicated tables have the same key definition, number of columns, column names, and column data types. If you define the replication scheme with the `TABLE DEFINITION CHECKING EXACT` clause, then the column definitions of replicated tables participating in the replication scheme must be identical on the master and subscriber databases. TimesTen recommends that you use

`TABLE DEFINITION CHECKING RELAXED`. For more information, see "Column definition options for replicated tables" in the *Oracle TimesTen In-Memory Database Replication Guide*.

***Example 4–12 Check consistency between replicated tables***

This output from the user error log shows a mismatch on the number of columns for the subscriber table `TTUSER.MYDSN`.

```
11:37:58.00 Info: REP:  9430: REP1:transmitter.c(4936): TT16136: Sending
definition for table TTUSER.MYDSN (1 column)
11:37:58.00 Info: REP:  9412: REP2:receiver.c(5928): TT16193: Adding definition
for table: TTUSER.MYDSN
11:37:58.00 Info: REP:  9412: REP2:meta.c(5580):TTUSER.MYDSN ptn 0: srcoff 0,
destoff 0, length 8
11:37:58.00 Info: REP:  9412: REP2:meta.c(5580):TTUSER.MYDSN ptn 1: srcoff 8,
destoff 12, length 12
11:37:58.00 Err : REP:  9412: REP2:receiver.c(6203): TT16198: Table definition
mismatch on number of columns for table TTUSER.MYDSN.  Local definition: 2;
transmitting peer: 1
11:37:58.00 Err : REP:  9412: REP2:receiver.c(6380): TT16204: Table TTUSER.MYDSN
marked invalid. Will not apply transactions received for it until a valid
definition is received
11:37:58.00 Err : REP:  9412: REP2:receiver.c(7200): TT16078: Table definition
for ID 637068 is invalid (Original failure 11:37:58
REP2:receiver.c(6203): TT16198: Table definition mismatch on number of columns
for table TTUSER.MYDSN.  Local definition: 2; transmitting peer: 1)
11:37:58.00 Err : REP:  9412: REP2:receiver.c(5002): TT16187: Transaction
1173958671/2; Error: transient 0, permanent 1
```

# Replication unresponsive, appears hung

Table summary is in the first heading cell.

| Possible cause | See... |
|---|---|
| There is a failed subscriber. | "Check replication state", below |
| Return-receipt timeout period is too long. | "Check return receipt timeout setting", below |

## Check replication state

Use the `ttReplicationStatus` procedure to check state of the subscriber database with respect to its master. If the subscriber is in the `Failed` state, see "Managing Database Failover and Recovery" in the *Oracle TimesTen In-Memory Database Replication Guide* for information on how to deal with failed databases.

***Example 4–13 Check replication state***

Use `ttReplicationStatus` to obtain the status of the `subscriberds` database from its master database, `masterDSN`, enter:

```
> ttIsql masterDSN
Command> CALL ttReplicationStatus ('subscriberds');
< SUBSCRIBERDS, MYHOST, 0, failed, 1, 10, REPSCHEME, REPL >
1 row found.
```

### Check return receipt timeout setting

Use the `ttRepSyncGet` procedure to check the return receipt timeout setting. A value of -1 indicates the application is to wait until it receives an acknowledgment from the subscriber. Network latency or other issues might delay receipt of the subscriber acknowledgment. You either address these issues or use the `ttRepSyncGet` procedure to reset the return receipt timeout period.

See "Check the status of return service transactions" in the *Oracle TimesTen In-Memory Database Replication Guide* for more information.

## Poor replication or XLA performance

Most of this section addresses issues that may impact replication performance. Some issues, such as log buffer too small and reading from the transaction log files on the file system, can impact the performance of both replication and XLA applications.

| Possible cause | See... |
|---|---|
| Network is slow. | "Check network bandwidth", below |
| `RETURN RECEIPT` is used. | "Check use of return receipt blocking", below |
| Replication scheme is inconsistent. | "Check replication configuration" on page 4-11 |
| Log buffer is too small. | "Check size of log buffer" on page 4-11 |
| There are frequent or inefficient file system writes. | "Check durability settings" on page 4-11 |
| Reading from transaction log files on the file system rather than the log buffer. | "Check for reads from transaction log files" on page 4-11 |
| Rate of conflicts is high. | "Conflict reporting slows down replication" on page 4-16 |

### Check network bandwidth

Replication agents typically communicate over some type of network connection. If replicating over a network slower than 10 MB per second (such as common with a 100 Base-T Ethernet typical in a LAN), you must be careful to match the transaction load to the available bandwidth of the network. see "Network bandwidth requirements" in the *Oracle TimesTen In-Memory Database Replication Guide* for details.

### Check use of return receipt blocking

Unless you need receipt confirmation for all your transactions, disable `RETURN RECEIPT BLOCKING`. If you require receipt confirmation for some transactions, then set `RETURN RECEIPT BY REQUEST` and call the `ttRepSyncSet` procedure to enable the return receipt service for specific transactions. See "RETURN RECEIPT BY REQUEST" in the *Oracle TimesTen In-Memory Database Replication Guide* for details.

**Note:** The performance degradation caused by return-receipt becomes less of an issue when multiple applications (or threads) are updating the database. If you must use return-receipt in a transaction, you can improve the performance of your application by using multiple threads to update the database. Though each thread must block for receipt confirmation, the other threads are free to make updates.

## Check replication configuration

In addition to return-receipt setting described above, other factors related to the configuration of your replication scheme could impact replication performance. As described in "Making decisions about performance and recovery tradeoffs" in the *Oracle TimesTen In-Memory Database Replication Guide*, you often have to weigh the ability to efficiently failover and recover a database against replication performance.

For more information about direct replication, see "Direct replication or propagation" in the *Oracle TimesTen In-Memory Database Replication Guide*.

## Check size of log buffer

Setting your log buffer too small may impact replication performance. Instead, Set the `LogBufMB` DSN attribute to a larger size. For more information on this DSN attribute, see "Setting connection attributes for logging" in the *Oracle TimesTen In-Memory Database Replication Guide*

## Check durability settings

You can improve replication performance by setting `TRANSMIT NONDURABLE` on the replication `ELEMENT` to eliminate the flush-log-to-file system operation from the replication cycle. See "Setting transmit durability on DATASTORE element" in the *Oracle TimesTen In-Memory Database Replication Guide* for details.

Enabling the `DURABLE COMMIT` option in your replication scheme also impacts performance. See "DURABLE COMMIT" in the *Oracle TimesTen In-Memory Database Replication Guide* for more information.

## Check for reads from transaction log files

In some situations a "log reader," such as a master replication agent 'transmitter' thread or a `ttXlaNextUpdate` call in an XLA application, may not be able to keep up with the update rate of the applications writing to the database. Normally, replication and XLA readers get update records from the log buffer in memory. When the readers fall behind the application update rate, transaction log files can accumulate on the file system until the backlog can be cleared. This forces the readers to read transactions from the transaction log files on the file system, which is much slower. Should you detect reads from the transaction log files, you may want to respond by decreasing the rate of application updates to that sustainable by the log readers.

Applications can monitor whether log readers are obtaining update records from transaction log files on the file system rather than from the log buffer in memory by tracking the `SYS.MONITOR` table entry `LOG_FS_READS`. For example, you can check the value of `LOG_FS_READS` for the database, `MASTERDSN`, with the following `ttIsql` command.

```
% ttIsql -v1 -e "select log_fs_reads from monitor; quit;" -connStr dsn=MASTERDSN
```

If the `LOG_FS_READS` counter is increasing, the log readers are falling behind or clearing out a backlog in the transaction log files.

For more complete monitoring of replication progress, create a simple shell script like the following:

```
!/bin/sh
trap exit 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
DSN=$1

while [ 1 ] ; do
   date
   ttRepAdmin -receiver -list -connStr dsn=$DSN
   echo -n "Log reads from disk: "
   ttIsql -v1 -e "select log_fs_reads from monitor; quit;" -connStr dsn=$DSN
   echo
   ttRepAdmin -bookmark -connStr dsn=$DSN
sleep 15
done
```

***Example 4–14 Check the status of the transaction log***

For example, you name the above script `monitorLog` and your replication scheme replicates from the `MASTERDSN` database to the `SUBSCRIBER1DSN` database. You can then check the status of the transaction log by entering:

```
% monitorLog masterdsn
```

This generates output similar to the following:

```
Mon Aug  2 10:44:40  2004
Peer name         Host name                Port    State  Proto
----------------  ------------------------ ------  ------- -----
SUBSCRIBER1DSN    MYHOST                   Auto    Start   12

Last Msg Sent Last Msg Recv Latency TPS     RecordsPS Logs
------------- ------------- ------- ------- --------- ----
00:00:05      -               -1.00      -1        -1    1

Log reads from disk: < 0 >

Replication hold LSN ...... 10/2656136
Last written LSN .......... 10/4015824
Last LSN forced to disk ... 10/3970152
```

The output from the script displays an updated status every 15 seconds until you enter Ctrl-C to exit.

Following the date in the output in Example 4–14 is the name of the subscriber, its host, and so on. Next is latency and rate information, as well as the number of transaction log files being retained on behalf of this subscriber. The specific meaning of each value is described in "Display subscriber status with the ttRepAdmin utility" in the *Oracle TimesTen In-Memory Database Replication Guide*. The main interest here is the 'Last Msg Sent' and 'Logs' values. The 'Last Msg Sent' value indicates the elapsed time since the last message was sent by the master to the subscriber and 'Logs' indicates how many transaction log files behind the replication log reader is from the current log insertion point used by the writers (Last written LSN).

Normally the 'Logs' value should be '1', as shown in Example 4–14. A steadily increasing 'Logs' value indicates latency is increasing and eventually log reads are satisfied from the file system.

---

**Note:** If the `LogBufMB` is larger than the `LogFileSize`, an increase in the 'Logs' value does not necessarily mean the log readers are reading from the transaction log files. This is because the log manager does not allow more than one log file's worth of data to be outstanding before writing it to the file system. After the log manager writes the data, the data remains in the log buffer to be read directly by the log readers. So, when the `LogBufMB` is larger than the `LogFileSize`, the 'Logs" value alone may not be the best measure of whether log readers are reading from memory or from the file system.

---

The output from the following command displays the number of the transaction log files and the location of the bookmarks set by the log manager.

```
ttRepAdmin -bookmark -connStr dsn=$DSN
```

For more information, see "Monitor replication with the ttRepAdmin utility" in the *Oracle TimesTen In-Memory Database Replication Guide*. The difference between the Replication hold LSN and the last written LSN indicates the number of records in the transaction log that have not yet been transmitted to the subscribers. A steady increase in the difference between these values is another indication that replication latency is increasing and log file reads are likely to occur.

***Example 4–15 Log reader must read from transaction log files***

In this example, assume the `LogBufMB` is 16MB and the `LogFileSize` is 8MB. The following output indicates the log reader is approximately 1.8 MB behind the capacity of the log buffer and must read from the transaction log files, 14 and 15.

```
Peer name         Host name                Port    State   Proto
----------------  ------------------------ ------  ------- -----
SUBSCRIBER1DSN    MYHOST                   Auto    Start   12

Last Msg Sent Last Msg Recv Latency TPS     RecordsPS Logs
------------- ------------- ------- ------- --------- ----
00:00:03      -               -1.00      -1        -1    4

Log reads from disk: <20>

Replication hold LSN ...... 14/7007464
Last written LSN .......... 17/465336
Last LSN forced to disk ... 17/456152
```

# Problems using ttRepAdmin

This section includes the following topics:

- Problems when changing the state of a replication receiver
- Problems using ttRepAdmin -duplicate
- Returns 'Must specify -scheme' error

## Problems when changing the state of a replication receiver

If you try to change the state of a replication receiver when replication has been configured to use a secondary IP address, a misconfiguration of the `/etc/hosts` file may cause `ttRepAdmin` to print the following error:

```
Alter replication with 'ALTER REPLICATION...port 0' failed:
TT0907: Unique constraint (REPSTORESIX) violated.
```

This error is caused by replication not recognizing the local database, which can be confirmed by the following query:

```
SELECT * FROM ttrep.ttstores WHERE is_local_store <> 0x0;
```

If this query returns no rows, or returns a row with the main host name of the database set to the result of the hostname command rather than the host you specified, you have encountered a configuration problem with the `/etc/hosts` file.

To correct the problem, make sure that the special host name you are using is defined in the `/etc/hosts` file and that there is an IP address in common between your special host name and the result of the `hostname` command.

For example, if the `hostname` command returns `softswitch` and your system has two Ethernet cards with the addresses `10.10.15.136` and `192.168.15.136`, the IP addresses defined for `softswitch` should include both IP addresses.

## Problems using ttRepAdmin -duplicate

Make sure that Timesten is installed with the same instance administrator on both the source system and target system. If so, then consider other causes:

- Database created before duplication
- Wrong database or host name

### Database created before duplication

If you connected to your new subscriber DSN before running `ttRepAdmin -duplicate`, the database has already been created. In this situation, `-duplicate` returns:

```
Error : Restore not done : The datastore already exists.
Unable to restore datastore locally
```

Confirm the existence of the database by running `ttStatus` and checking to see if the database is in the returned list. If the new subscriber database exists, destroy it and try `ttRepAdmin -duplicate` again.

```
> ttDestroy /tmp/newstore
> ttRepAdmin -dsn newstoreDSN -duplicate -name newstore
-from masterds -host "server1"
```

### Wrong database or host name

If you have made an error entering the subscriber database name or host name in the replication scheme, you may see something like the following:

```
Unable to swap datastore locally
No receiver NEWSTORE on SERVER2 found to swap with
```

If you provide an incorrect host name for a subscriber, you may receive this error:

```
Problem: ttRepAdmin -duplicate command returns TimesTen error 12080: No
subscriber found to swap with.
```

You also receive information about the subscriber that TimesTen is trying to locate. One common cause is providing an incorrect host name, which must be the exact same name as the host name provided when issuing the `CREATE ACTIVE STANDBY PAIR` statement. For example, if you created the subscriber with `myhost.oracle.com`, but only provided `myhost` in the `ttRepAdmin -duplicate`, the subscriber will not be found.

> **Note:** If `myhost` is the local host, use the `-localhost` argument. You typically need to use the `-localhost` argument if the local host name does not exactly match the host name provided when creating the replication scheme.

## Returns 'Must specify -scheme' error

If you have more than one scheme specified in your `TTREP.REPLICATIONS` table, some `ttRepAdmin` commands may return the error:

```
Must specify -scheme to identify which replication scheme to use
```

To check the names of the replication schemes used by your database, use the `ttIsql` utility to connect, and enter:

```
Command> SELECT * from TTREP.REPLICATIONS;
```

***Example 4–16 Two replication schemes assigned to the database***

This example shows that two replication schemes, `REPSCHEME1` and `REPSCHEME2`, are assigned to the database associated with `subDSN`. In this case, it is necessary to use the `ttRepAdmin -scheme` option.

```
> ttIsql -connStr "dsn=subDSN"
Command> SELECT * from TTREP.REPLICATIONS;
< REPSCHEME1      , REPL                , C, 0, 0, -1 >
< REPSCHEME2      , REPL                , C, 0, 0, -1 >
2 rows found.
Command> exit
> ttRepAdmin -dsn subDSN -receiver -list -scheme REPSCHEME1
Peer name         Host name                 Port    State  Proto
----------------  ------------------------ ------  ------- -----
SUBSCRIBER1        MYHOST                   Auto   Start      10

Last Msg Sent Last Msg Recv Latency TPS     RecordsPS Logs
------------- ------------- ------- ------- --------- ----
0:01:12       -               19.41       5        52    2
```

# Problems with conflict checking

This section includes the following topics:

- Column cannot be used for replication timestamp
- Timestamp does not exist
- Conflict reporting slows down replication

## Column cannot be used for replication timestamp

When attempting to set `CHECK CONFLICTS` for an element in a `CREATE REPLICATION` statement, you may encounter an error similar to the following.

```
8004: Column REPL.TABS.TS cannot be used for replication timestamp checking if
in an index or added by ALTER TABLE; and must be binary(8) with NULL values
allowed.
```

In this situation, check:

- That the timestamp column in the specified table is a nullable column of type `BINARY(8)`. In the above example, the TS column in the `REPL.TAB` table should have a type of `BINARY(8)`.
- The timestamp column is defined in the original `CREATE TABLE` statement, rather than added later using `ALTER TABLE`.

### Timestamp does not exist

You may receive an error similar to the following:

```
2208: Column TS does not exist in table.
```

In this situation, confirm that you have specified the correct name for the timestamp column in the `CHECK CONFLICTS` clause and that it exists in the specified table.

Also, make sure the timestamp column is not part of a primary key or index.

### Conflict reporting slows down replication

If there are many conflicts in a short period of time, subscriber performance can slow down because of the reporting requirements. You can use store attributes in the `CREATE REPLICATION` or `ALTER REPLICATION` statements to suspend and resume conflict reporting at specified rates of conflict:

- `CONFLICT REPORTING SUSPEND AT` *`rate`*
- `CONFLICT REPORTING RESUME AT` *`rate`*

Information about conflicts that occur while reporting is suspended cannot be retrieved.

See "Reporting conflicts" in the *Oracle TimesTen In-Memory Database Replication Guide*.

## Problems with parallel replication

If you have configured parallel replication, then you have additional TimesTen replication threads running on your system. Be sure that you have enough CPU cores to accommodate the increased number of replication threads.

If you have configured user-defined track based parallel replication (`ReplicationApplyOrdering` = 1 and `ReplicationParallelism` > = 2), and you are seeing permanent errors on the receiver side that include constraint violations or row lookup failures, then possible causes include:

- Application defined dependencies are incorrect
- Operations for the same keys or keys of tables in a constraint relationship are written to different tracks.

Check the value of the general connection attribute, `ReplicationTrack`. All transactions executed by the connection are assigned to the track as defined in the `ReplicationTrack` attribute setting. Make sure the workload for each track is suitable. Use the `ALTER SESSION` statement to change the replication track. For more information, see "ALTER SESSION" in the *Oracle TimesTen In-Memory Database SQL*

*Reference*. For more information on configuring parallel replication, see "Configuring parallel replication" in the *Oracle TimesTen In-Memory Database Replication Guide*.

# 5

# Troubleshooting AWT Cache Groups

Creating an asynchronous writethrough (AWT) cache group automatically creates a replication scheme that enables the database to communicate with the Oracle database. You must start the replication agent after you create an AWT cache group and before you start the cache agent. See "Creating an AWT cache group" in the *Oracle TimesTen Application-Tier Database Cache User's Guide* for more details.

This chapter contains the following troubleshooting sections:

- Replication agent not started
- AWT performance monitoring
- Possible causes of poor AWT performance
- Oracle Database errors reported by TimesTen for AWT

## Replication agent not started

The replication agent must be started in order for the AWT cache group to work. If the replication agent is not starting, use the information in Chapter 4, "Troubleshooting Replication" to troubleshoot the replication agent.

If you are unable to get replication working, the problem may solved by one or more of the following sections:

| Possible Cause | See... |
|---|---|
| Replication agent will not start. | "Unable to start or stop replication agent" on page 4-2 |
| TimesTen daemon or replication agents are not running. | "Check status of TimesTen daemon and replication agents" on page 4-3 |
| Replication agents are not communicating. | "Check that replication agents are communicating" on page 4-5 |
| Replication is not in Start state. | "Check replication state" on page 4-5 |

## AWT performance monitoring

You can monitor the performance of asynchronous writethrough (AWT) cache groups to determine how much time is spent performing tasks in the AWT workflow.

The following sections describe how to enable the monitoring and the tools for viewing the results.

- Enable AWT performance monitoring

- Display AWT performance results with the ttRepAdmin utility
- Using system tables to monitor AWT operations

## Enable AWT performance monitoring

Use the `ttCacheAWTMonitorConfig` built-in procedure to enable monitoring for AWT cache groups. Ensure that the replication agent is running before calling `ttCacheAWTMonitorConfig`.

The following example enables monitoring and sets the sampling frequency to 16. A sampling factor of 16 is recommended for accuracy and performance.

```
Command> CALL ttCacheAWTMonitorConfig('ON',16);
< ON, 16 >
1 row found.
```

## Display AWT performance results with the ttRepAdmin utility

Use the `ttRepAdmin` utility with the `-awtmoninfo` and `-showstatus` commands to display the monitoring results. The AWT monitoring statistics include the following:

- TimesTen processing time: The total number of milliseconds spent in processing AWT transaction data since monitoring was enabled.
- Oracle Database bookmark management time: The total number of milliseconds spent in managing AWT metadata on Oracle Database since monitoring was enabled.
- Oracle Database execute time: The total number of milliseconds spent in OCI preparation, binding and execution for AWT SQL operations since monitoring was enabled. This statistic includes network latency between TimesTen and Oracle Database.
- Oracle Database commit time: The total number of milliseconds spent in committing AWT updates on Oracle Database since monitoring was enabled. This statistic includes network latency between TimesTen and Oracle Database.
- Time since monitoring was started.
- Total number of TimesTen row operations: The total number of rows updated in AWT cache groups since monitoring was enabled.
- Total number of TimesTen transactions: The total number of transactions in AWT cache groups since monitoring was enabled.
- Total number of flushes to the Oracle database: The total number of times that TimesTen data has been sent to the Oracle database.
- The percentage of time spent on TimesTen processing, Oracle Database bookmark management, Oracle Database execution and Oracle Database commits.

For example:

```
ttRepAdmin -showstatus -awtmoninfo myDSN

[other -showstatus output]
...
AWT Monitoring statistics
-------------------------
 TimesTen processing time : 8443.424000 millisecs (44.064958 )
   Oracle execute  (SQL execution) time : 8930.320000 millisecs (46.605994 )
   Oracle execute (PL/SQL execution) time : 0.000000 millisecs (0 )
   Oracle commit time : 1787.568000 millisecs (9.329048 )
   Time since monitoring was started: 21954.410000 millisecs
   CacheAwtMethod mode : 0
   Cache-connect Operational Stats :
      SQL Operations sent to Oracle : 143556
           Number of update operations : 58
           Number of update batches    : 58
           Number of insert operations : 143498
           Number of insert batches    : 1146
           Number of delete operations : 0
           Number of delete batches    : 0
           Total number of batches sent: 1204
           Number of bytes sent : 4769094
      Number of TimesTen Transactions sent to Oracle (includes retries) :
            143556
      Number of retries on TimesTen due to errors on Oracle : 0
      Number of round trips to Oracle (includes executes, commits and
            rollbacks) : 2290
      Number of commits on Oracle : 1086
      Number of rollbacks on Oracle : 0
      Number of rxbatches: 244
      Number of rxskips: 0

Peer name         Host name                Port    State  Proto
----------------  ------------------------ ------  ------- -----
_ORACLE           MYHOST                   Auto   Start      27

Last Msg Sent Last Msg Recv Latency TPS     RecordsPS Logs
------------- ------------- ------- ------- --------- ----
00:00:00      -                2.35    6311      6311    1
```

## Using system tables to monitor AWT operations

To determine whether asynchronous writethrough operations are keeping up with the rate of updates on cache tables in AWT cache groups, query the `LAST_LOG_FILE`, `REPHOLD_LOG_FILE` and `REPHOLD_LOG_OFF` columns of the `SYS.MONITOR` system table. If the difference between the value in the `LAST_LOG_FILE` column and the value in the `REPHOLD_LOG_FILE` column is increasing over time, and the value in the `REPHOLD_LOG_OFF` column is increasing slowly or not changing, then the tables are being updated at a faster rate than the updates are being replicated.

Then run a `ttRepAdmin -receiver -list` utility command and find the row where `_ORACLE` is in the `Peer name` field. View the values in the `Last Msg Sent`, `Last Msg Recv`, `Latency` and `TPS` fields within the same row to determine if the replication activity that is falling behind is asynchronous writethrough operations.

For more information about monitoring log files, see "Monitoring accumulation of transaction log files" in *Oracle TimesTen In-Memory Database Operations Guide*.

The `SYS.SYSTEMSTATS` system table has a variety of statistics about AWT performance. See "SYS.SYSTEMSTATS" in *Oracle TimesTen In-Memory Database System Tables and Views Reference*.

# Possible causes of poor AWT performance

This section addresses issues that may degrade AWT cache group performance.

| Possible cause | See... |
|---|---|
| Network is slow. | "Check network bandwidth" on page 4-10 |
| Log buffer is too small. | "Check size of log buffer" on page 4-11 |
| There are frequent or inefficient file system writes. | "Check durability settings" on page 4-11 |
| Reading from transaction log files on the file system instead of the log buffer. | "Check for reads from transaction log files" on page 4-11 |
| There are problems on the Oracle database. | "Problems on the Oracle Database", below |
| Replication agent is down. | "Unable to start or stop replication agent" on page 4-2 or "Replication does not work" on page 4-3 |
| `UNLOAD CACHE GROUP` cannot complete if row updates are not propagated to the Oracle database. | "UNLOAD CACHE GROUP requires row updates to complete propagation", below |

### Problems on the Oracle Database

If an AWT cache group cannot propagate updates to the Oracle database, the causes can include network failure, an Oracle database failure, I/O bottlenecks and CPU bottlenecks.

AWT propagation to the Oracle database can also fail because the replication agent is stopped. See "Unable to start or stop replication agent" on page 4-2 and "Replication does not work" on page 4-3.

### UNLOAD CACHE GROUP requires row updates to complete propagation

Execution of the `UNLOAD CACHE GROUP` statement for an AWT cache group waits until updates on the rows have been propagated to the Oracle database. However, the `UNLOAD CACHE GROUP` statement may fail if data propagation to the Oracle database is stopped for any reason, such as network failure, the Oracle database is down, or the replication agent is stopped.

## Oracle Database errors reported by TimesTen for AWT

The following sections describe both permanent and transient Oracle Database errors that can be reported by TimesTen for an AWT cache group:

- Permanent Oracle Database errors reported by TimesTen
- Transient Oracle Database errors reported by TimesTen

### Permanent Oracle Database errors reported by TimesTen

Insert, update, or delete errors that occur while applying changes to Oracle Database are saved in an error file located in the database directory with the following name:

```
DataSourceName.awterrs
```

Errors reported to this file are *permanent* errors. TimesTen does not retry the transaction. The errors may be reported in the AWT error file long after the commit to TimesTen occurs.

The format of the messages in the AWT error file is similar to those generated for conflict and transaction errors in replication, as shown in Example 5–1. Oracle

Database error messages are also reported in the support log and the user log. The Oracle Database errors are prefixed with 'ORA-.' The TimesTen error messages are prefixed with 'TT.' For example, in Example 5–1, TimesTen errors 5210 and 5025 are shown as TT5210 and TT5025. These can be looked up in the *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps* guide.

***Example 5–1 Cache violation occurs when update is propagated to Oracle Database***

If a constraint violation occurs when a cache group update is propagated to the Oracle database, the message in the AWT error file is similar to the following:

```
Error occurred 14:48:55 on 03-22-2011
Datastore: c:\temp\cgDSN
Oracle Id: system1
Transmitting name: cgDSN
Error message:
  TT5210: Oracle unique constraint violation error in OCIStmtExecute():
ORA-00001: unique constraint (GUSER.SYS_C00357240) violated rc = -1 -- file
 "bdbTblH.c", lineno 1205, procedure "ttBDbStmtForce()"
  TT5025: Commit failure in Oracle. Transaction must be rolled back in TimesTen.
-- file "bdbConnect.c", lineno 885, procedure "ttBDbXact()"

Operation that caused the error:
Insert into table TESTUSER.T1 <9,1000>

Failed transaction:
Insert into table TESTUSER.T1 <9, 1000>
End of failed transaction
```

***Example 5–2 An object that TimesTen has placed on the Oracle database is dropped***

If an object that TimesTen has placed on the Oracle database is dropped, the message in the AWT error file is similar to the following:

```
May 04 18:12:36 HOST1 TimesTen Replication 11.2[2136]:
[Err ] DEFAULT:meta.c(639):
TT16062: Failed to compile command:
select p.commit_timestamp, p.commit_seqnum, p.protocol from owner1.TT_06_REPPEERS
p where p.replication_name = :rname and p.replication_owner = :rowner and
p.tt_store_id = :oid and p.subscriber_id = :sid

May 04 18:12:36 HOST1 TimesTen Replication 11.2[2136]:
[Err ] DEFAULT:meta.c(639):
TT5221: TT5221: Oracle syntax error in OCIStmtExecute():
ORA-00942: table or view does not exist rc = -1 -- file "bdbStmt.c", lineno 1041,
procedure "getOraOutTypesNLengths()"
```

In this example, the `TT_06_REPPEERS` table does not exist. To recover from this error, perform the following tasks:

1. Stop the replication agent.
2. Drop and recreate the cache group.
3. Restart the replication agent.

## Transient Oracle Database errors reported by TimesTen

The support log for databases with AWT cache groups may contain Oracle Database errors if the replication agent encounters a problem on the Oracle database. If the replication agent encounters one of these errors, AWT rolls back the transaction and retries it. If the support log becomes full, the oldest messages are deleted and replaced by new messages.

The Oracle Database errors in the support log are considered *transient* because AWT retries the transaction.

Some transient errors indicate an underlying problem on the Oracle database must be solved before AWT operations can continue. For example:

```
ORA-01536: space quota exceeded for tablespace
ORA-01034: ORACLE not available
```

After the underlying problem has been fixed, AWT retries the operation.

The following Oracle Database errors are transient:

```
ORA-00018: maximum number of sessions exceeded
ORA-00019: maximum number of session licenses exceeded
ORA-00020: maximum number of processes (%s) exceeded
ORA-00025: failed to allocate %s
ORA-00028: your session has been killed
ORA-00038: Cannot create session: server group belongs to another user
ORA-00051: timeout occurred while waiting for a resource
ORA-00052: maximum number of enqueue resources (%s) exceeded
ORA-00053: maximum number of enqueues exceeded
ORA-00054: resource busy and acquire with NOWAIT specified
ORA-00055: maximum number of DML locks exceeded
ORA-00057: maximum number of temporary table locks exceeded
ORA-00058: DB_BLOCK_SIZE must be %s to mount this database (not %s)
ORA-00059: maximum number of DB_FILES exceeded
ORA-00060: deadlock detected while waiting for resource
ORA-00063: maximum number of LOG_FILES exceeded
ORA-00064: object is too large to allocate on this O/S (%s,%s)
ORA-00099: timed out while waiting for resource, potential PDML deadlock
ORA-00104: deadlock detected; all public servers blocked waiting for resources
ORA-00107: failed to connect to ORACLE listener process
ORA-00115: connection refused; dispatcher connection table is full
ORA-00125: connection refused; invalid presentation
ORA-00126: connection refused; invalid duplicity
ORA-00284: recovery session still in progress
ORA-00370: potential deadlock during kcbchange operation
ORA-00371: not enough shared pool memory
ORA-00376: file %s cannot be read at this time
ORA-00379: no free buffers available in buffer pool %s for block size %sK
ORA-00384: Insufficient memory to grow cache
ORA-00568: Maximum number of interrupt handlers exceeded
ORA-00579: osndnt: server received malformed connection request
ORA-00600: internal error code, arguments: [%s], [%s], [%s], [%s], [%s], [%s],
[%s], [%s]
ORA-00603: ORACLE server session terminated by fatal error
ORA-01000: maximum open cursors exceeded
ORA-01012: not logged on
ORA-01014: ORACLE shutdown in progress
ORA-01019: unable to allocate memory in the user side
ORA-01031: insufficient privileges
ORA-01033: ORACLE initialization or shutdown in progress
ORA-01034: ORACLE not available
ORA-01035: ORACLE only available to users with RESTRICTED SESSION privilege
ORA-01037: maximum cursor memory exceeded
ORA-01046: cannot acquire space to extend context area
ORA-01073: fatal connection error: unrecognized call type
ORA-01089: immediate shutdown in progress - no operations are permitted
```

```
ORA-01090: shutdown in progress - connection is not permitted
ORA-01092: ORACLE instance terminated. Disconnection forced
ORA-01094: ALTER DATABASE CLOSE in progress. Connections not permitted
ORA-01109: database not open
ORA-01147: SYSTEM tablespace file %s is offline
ORA-01154: database busy. Open, close, mount, and dismount not allowed now
ORA-01155: the database is being opened, closed, mounted or dismounted
ORA-01219: database not open: queries allowed on fixed tables/views only
ORA-01237: cannot extend datafile %s
ORA-01456: may not perform insert/delete/update operation inside a READ ONLY
transaction
ORA-01536: space quota exceeded for tablespace '%s'
ORA-01539: tablespace '%s' is not online
ORA-01542: tablespace '%s' is offline, cannot allocate space in it
ORA-01562: failed to extend rollback segment number %s
ORA-01573: shutting down instance, no further change allowed
ORA-01628: max # extents (%s) reached for rollback segment %s
ORA-01629: max # extents (%s) reached saving undo for tablespace %s
ORA-01630: max # extents (%s) reached in temp segment in tablespace %s
ORA-01631: max # extents (%s) reached in table %s.%s
ORA-01632: max # extents (%s) reached in index %s.%s
ORA-01650: unable to extend rollback segment %s by %s in tablespace %s
ORA-01651: unable to extend save undo segment by %s for tablespace %s
ORA-01652: unable to extend temp segment by %s in tablespace %s
ORA-01653: unable to extend table %s.%s by %s in tablespace %s
ORA-01654: unable to extend index %s.%s by %s in tablespace %s
ORA-01655: unable to extend cluster %s.%s by %s in tablespace %s
ORA-01656: max # extents (%s) reached in cluster %s.%s
ORA-01658: unable to create INITIAL extent for segment in tablespace %s
ORA-01659: unable to allocate MINEXTENTS beyond %s in tablespace %s
ORA-01680: unable to extend LOB segment by %s in tablespace %s
ORA-01681: max # extents (%s) reached in LOB segment in tablespace %s
ORA-01683: unable to extend index %s.%s partition %s by %s in tablespace %s
ORA-01684: max # extents (%s) reached in table %s.%s partition %s
ORA-01685: max # extents (%s) reached in index %s.%s partition %s
ORA-01686: max # files (%s) reached for the tablespace %s
ORA-01688: unable to extend table %s.%s partition %s by %s in tablespace %s
ORA-01691: unable to extend lob segment %s.%s by %s in tablespace %s
ORA-01692: unable to extend lob segment %s.%s partition %s by %s in tablespace %s
ORA-01693: max # extents (%s) reached in lob segment %s.%s
ORA-01694: max # extents (%s) reached in lob segment %s.%s partition %s
ORA-03113: end-of-file on communication channel
ORA-03114: not connected to ORACLE
ORA-03134: Connections to this server version are no longer supported.
ORA-03135: connection lost contact
ORA-03136: inbound connection timed out
ORA-03232: unable to allocate an extent of %s blocks from tablespace %s
ORA-03233: unable to extend table %s.%s subpartition %s by %s in tablespace %s
ORA-03234: unable to extend index %s.%s subpartition %s by %s in tablespace %s
ORA-03235: max # extents (%s) reached in table %s.%s subpartition %s
ORA-03236: max # extents (%s) reached in index %s.%s subpartition %s
ORA-03237: Initial Extent of specified size cannot be allocated
ORA-03238: unable to extend LOB segment %s.%s subpartition %s by %s in tablespace
%s
ORA-03239: maxextents (%s) reached in LOB segment %s.%s subpartition %s
ORA-04020: deadlock detected while trying to lock object %s%s%s%s%s
ORA-06019: NETASY: invalid login (connect) string
ORA-06021: NETASY: connect failed
ORA-06030: NETDNT: connect failed, unrecognized node name
ORA-06031: NETDNT: connect failed, unrecognized object name
```

```
ORA-06032: NETDNT: connect failed, access control data rejected
ORA-06033: NETDNT: connect failed, partner rejected connection
ORA-06034: NETDNT: connect failed, partner exited unexpectedly
ORA-06035: NETDNT: connect failed, insufficient resources
ORA-06036: NETDNT: connect failed, no response from object
ORA-06037: NETDNT: connect failed, node unreachable
ORA-06039: NETDNT: connect failed
ORA-06040: NETDNT: invalid login (connect) string
ORA-06108: NETTCP: connect to host failed
ORA-06113: NETTCP: Too many connections
ORA-06114: NETTCP: SID lookup failure
ORA-06143: NETTCP: maximum connections exceeded
ORA-06315: IPA: Invalid connect string
ORA-06316: IPA: Invalid database SID
ORA-06317: IPA: Local maximum number of users exceeded
ORA-06318: IPA: Local maximum number of connections exceeded
ORA-06319: IPA: Remote maximum number of users exceeded
ORA-06320: IPA: Remote maximum number of connections exceeded
ORA-06404: NETCMN: invalid login (connect) string
ORA-06413: Connection not open.
ORA-10435: enable tracing of global enqueue service deadlock detetction
ORA-10626: specify timeout for online index rebuild to wait for DML
ORA-10906: Unable to extend segment after insert direct load
ORA-12150: TNS:unable to send data
ORA-12151: TNS:received bad packet type from network layer
ORA-12152: TNS:unable to send break message
ORA-12153: TNS:not connected
ORA-12154: TNS:could not resolve service name
ORA-12155: TNS:received bad datatype in NSWMARKER packet
ORA-12156: TNS:tried to reset line from incorrect state
ORA-12157: TNS:internal network communication error
ORA-12158: TNS:could not initialize parameter subsystem
ORA-12159: TNS:trace file not writeable
ORA-12160: TNS:internal error: Bad error number
ORA-12161: TNS:internal error: partial data received
ORA-12162: TNS:service name is incorrectly specified
ORA-12163: TNS:connect descriptor is too long
ORA-12166: TNS:Client can not connect to HO agent.
ORA-12168: TNS:Unable to contact Directory Server.
ORA-12169: TNS:Net service name given as connect identifier is too long
ORA-12170: TNS:Connect timeout occurred
ORA-12171: TNS:could not resolve connect identifier: %s
ORA-12196: TNS:received an error from TNS
ORA-12197: TNS:keyword-value resolution error
ORA-12198: TNS:could not find path to destination
ORA-12200: TNS:could not allocate memory
ORA-12201: TNS:encountered too small a connection buffer
ORA-12202: TNS:internal navigation error
ORA-12203: TNS:unable to connect to destination
ORA-12204: TNS:received data refused from an application
ORA-12205: TNS:could not get failed addresses
ORA-12206: TNS:received a TNS error during navigation
ORA-12207: TNS:unable to perform navigation
ORA-12208: TNS:could not find the TNSNAV.ORA file
ORA-12209: TNS:encountered uninitialized global
ORA-12210: TNS:error in finding Navigator data
ORA-12211: TNS:needs PREFERRED_CMANAGERS entry in TNSNAV.ORA
ORA-12212: TNS:incomplete PREFERRED_CMANAGERS binding in TNSNAV.ORA
ORA-12213: TNS:incomplete PREFERRED_CMANAGERS binding in TNSNAV.ORA
ORA-12214: TNS:missing local communities entry in TNSNAV.ORA
```

```
ORA-12215: TNS:poorly formed PREFERRED_NAVIGATORS Addresses in TNSNAV.ORA
ORA-12216: TNS:poorly formed PREFERRED_CMANAGERS addresses in TNSNAV.ORA
ORA-12217: TNS:could not contact PREFERRED_CMANAGERS in TNSNAV.ORA
ORA-12218: TNS:unacceptable network configuration data
ORA-12219: TNS:missing community name from address in ADDRESS_LIST
ORA-12221: TNS:illegal ADDRESS parameters
ORA-12222: TNS:no such protocol adapter
ORA-12223: TNS:internal limit restriction exceeded
ORA-12224: TNS:no listener
ORA-12225: TNS:destination host unreachable
ORA-12226: TNS:operating system resource quota exceeded
ORA-12227: TNS:syntax error
ORA-12228: TNS:protocol adapter not loadable
ORA-12229: TNS:Interchange has no more free connections
ORA-12230: TNS:Severe Network error ocurred in making this connection
ORA-12231: TNS:No connection possible to destination
ORA-12232: TNS:No path available to destination
ORA-12233: TNS:Failure to accept a connection
ORA-12235: TNS:Failure to redirect to destination
ORA-12236: TNS:protocol adapter not loaded
ORA-12316: syntax error in database link's connect string
ORA-12326: database %s is closing immediately;  no operations are permitted
ORA-12329: database %s is closed;  no operations are permitted
ORA-12500: TNS:listener failed to start a dedicated server process
ORA-12501: TNS:listener failed to spawn process
ORA-12502: TNS:listener received no CONNECT_DATA from client
ORA-12504: TNS:listener was not given the SID in CONNECT_DATA
ORA-12505: TNS:listener could not resolve SID given in connect descriptor
ORA-12506: TNS:listener was not given the ALIAS in CONNECT_DATA
ORA-12507: TNS:listener could not resolve ALIAS given
ORA-12508: TNS:listener could not resolve the COMMAND given
ORA-12509: TNS:listener failed to redirect client to service handler
ORA-12510: TNS:database temporarily lacks resources to handle the request
ORA-12511: TNS:service handler found but it is not accepting connections
ORA-12512: TNS:service handler found but it has not registered a
redirect address
ORA-12513: TNS:service handler found but it has registered for a
different protocol
ORA-12514: TNS:listener could not resolve SERVICE_NAME given in
connect descriptor
ORA-12515: TNS:listener could not find a handler for this presentation
ORA-12516: TNS:listener could not find available handler with matching
protocol stack
ORA-12517: TNS:listener could not find service handler supporting direct handoff
ORA-12518: TNS:listener could not hand off client connection
ORA-12519: TNS:no appropriate service handler found
ORA-12520: TNS:listener could not find available handler for requested
type of server
ORA-12521: TNS:listener could not resolve INSTANCE_NAME given in
connect descriptor
ORA-12522: TNS:listener could not find available instance with given
INSTANCE_ROLE
ORA-12523: TNS:listener could not find instance appropriate for the
client connection
ORA-12524: TNS:listener could not resolve HANDLER_NAME given in
connect descriptor
ORA-12525: TNS:listener has not received client's request in time allowed
ORA-12526: TNS:listener: all appropriate instances are in restricted mode
ORA-12527: TNS:listener: all instances are in restricted mode or blocking
new connections
```

```
ORA-12528: TNS:listener: all appropriate instances are blocking new connections
ORA-12529: TNS:connect request rejected based on current filtering rules
ORA-12531: TNS:cannot allocate memory
ORA-12532: TNS:invalid argument
ORA-12533: TNS:illegal ADDRESS parameters
ORA-12534: TNS:operation not supported
ORA-12535: TNS:operation timed out
ORA-12536: TNS:operation would block
ORA-12537: TNS:connection closed
ORA-12538: TNS:no such protocol adapter
ORA-12539: TNS:buffer over- or under-flow
ORA-12540: TNS:internal limit restriction exceeded
ORA-12541: TNS:no listener
ORA-12542: TNS:address already in use
ORA-12543: TNS:destination host unreachable
ORA-12544: TNS:contexts have different wait/test functions
ORA-12545: Connect failed because target host or object does not exist
ORA-12546: TNS:permission denied
ORA-12547: TNS:lost contact
ORA-12549: TNS:operating system resource quota exceeded
ORA-12550: TNS:syntax error
ORA-12551: TNS:missing keyword
ORA-12552: TNS:operation was interrupted
ORA-12554: TNS:current operation is still in progress
ORA-12555: TNS:permission denied
ORA-12556: TNS:no caller
ORA-12557: TNS:protocol adapter not loadable
ORA-12558: TNS:protocol adapter not loaded
ORA-12560: TNS:protocol adapter error
ORA-12561: TNS:unknown error
ORA-12562: TNS:bad global handle
ORA-12564: TNS:connection refused
ORA-12566: TNS:protocol error
ORA-12569: TNS:packet checksum failure
ORA-12570: TNS:packet reader failure
ORA-12571: TNS:packet writer failure
ORA-12574: TNS:redirection denied
ORA-12582: TNS:invalid operation
ORA-12583: TNS:no reader
ORA-12585: TNS:data truncation
ORA-12589: TNS:connection not bequeathable
ORA-12590: TNS:no I/O buffer
ORA-12591: TNS:event signal failure
ORA-12592: TNS:bad packet
ORA-12593: TNS:no registered connection
ORA-12595: TNS:no confirmation
ORA-12596: TNS:internal inconsistency
ORA-12600: TNS: string open failed
ORA-12602: TNS: Connection Pooling limit reached
ORA-12606: TNS: Application timeout occurred
ORA-12607: TNS: Connect timeout occurred
ORA-12608: TNS: Send timeout occurred
ORA-12609: TNS: Receive timeout occurred
ORA-12612: TNS:connection is busy
ORA-12615: TNS:preempt error
ORA-12623: TNS:operation is illegal in this state
ORA-12624: TNS:connection is already registered
ORA-12636: Packet send failed
ORA-12637: Packet receive failed
ORA-12829: Deadlock - itls occupied by siblings at block %s of file %s
```

```
ORA-12993: tablespace '%s' is offline, cannot drop column
ORA-14117: partition resides in offlined tablespace
ORA-14268: subpartition '%s' of the partition resides in offlined tablespace
ORA-16000: database open for read-only access
ORA-16003: standby database is restricted to read-only access
ORA-16403: shutdown in progress - remote connection is not permitted
ORA-16724: the intended state for resource has been set to OFFLINE
ORA-16903: Unable to connect to database
ORA-16914: Missing connect string. Try \"help\"
ORA-18014: deadlock detected while waiting for resource %s
ORA-21521: exceeded maximum number of connections in OCI (object mode only)
ORA-21522: attempted to use an invalid connection in OCI (object mode only)
ORA-23317: a communication failure has occurred
ORA-24401: cannot open further connections
ORA-24418: Cannot open further sessions.
ORA-24778: cannot open connections
ORA-25400: must replay fetch
ORA-25401: can not continue fetches
ORA-25402: transaction must roll back
ORA-25403: could not reconnect
ORA-25405: transaction status unknown
ORA-25407: connection terminated
ORA-25408: can not safely replay call
ORA-25409: failover happened during the network operation,cannot continue
ORA-25425: connection lost during rollback
ORA-29306: datafile %s is not online
ORA-30006: resource busy; acquire with WAIT timeout expired
ORA-30036: unable to extend segment by %s in undo tablespace '%s'
ORA-30040: Undo tablespace is offline
ORA-31443: deadlock detected while acquiring lock on %s
ORA-37013: (XSACQUIRE_DEADLOCK) Cannot wait to acquire object %j since doing so
would cause a deadlock.
ORA-44317: database open read-only
```

# Index