Oracle TimesTen In-Memory Database Troubleshooting Procedures Guide

Release 7.0

B31688-01

February 2007

Printed in the United States of America

Contents

3 Troubleshooting Cache Connect to Oracle

4 Troubleshooting Replication

Index

About this Guide

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

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

TimesTen documentation

TimesTen documentation is available on the product distribution media and on the Oracle Technology Network: http://www.oracle.com/technology/documentation/timesten_doc.html.

Including this guide, the TimesTen documentation set consists of these documents:

Book Titles	Description
Oracle TimesTen In-Memory Database Installation Guide	Contains information needed to install and configure TimesTen on all supported platforms.
Oracle TimesTen In-Memory Database Introduction	Describes all the available features in the Oracle TimesTen In-Memory Database.
Oracle TimesTen In-Memory Database Operations Guide	Provides information on configuring TimesTen and using the ttIsql utility to manage a data store. This guide also provides a basic tutorial for TimesTen.
Oracle TimesTen In-Memory Database C Developer's and Reference Guide and the *Oracle TimesTen In-Memory Database Java Developer's and Reference Guide*	Provide information on how to use the full set of available features in TimesTen to develop and implement applications that use TimesTen.
Oracle TimesTen In-Memory Database API Reference Guide	Describes all TimesTen utilities, procedures, APIs and provides a reference to other features of TimesTen.
Oracle TimesTen In-Memory Database SQL Reference Guide	Contains a complete reference to all TimesTen SQL statements, expressions and functions, including TimesTen SQL extensions.

Oracle TimesTen In-Memory Database Error Messages and SNMP Traps	Contains a complete reference to the TimesTen error messages and information on using SNMP Traps with TimesTen.
Oracle TimesTen In-Memory Database TTClasses Guide	Describes how to use the TTClasses C++ API to use the features available in TimesTen to develop and implement applications.
TimesTen to TimesTen Replication Guide	Provides information to help you understand how TimesTen Replication works and step-by-step instructions and examples that show how to perform the most commonly needed tasks. This guide is for application developers who use and administer TimesTen and for system administrators who configure and manage TimesTen Replication.
TimesTen Cache Connect to Oracle Guide	Describes how to use Cache Connect to cache Oracle data in TimesTen data stores. This guide is for developers who use and administer TimesTen for caching Oracle data.
Oracle TimesTen In-Memory Database Troubleshooting Procedures Guide	Provides information and solutions for handling problems that may arise while developing applications that work with TimesTen, or while configuring or managing TimesTen.

Background reading

For a Java reference, see:

- Horstmann, Cay and Gary Cornell. *Core Java(TM) 2, Volume I--Fundamentals (7th Edition) (Core Java 2)*. Prentice Hall PTR; 7 edition (August 17, 2004).

A list of books about ODBC and SQL is in the Microsoft ODBC manual included in your developer's kit. Your developer's kit includes the appropriate ODBC manual for your platform:

- *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide* provides all relevant information on ODBC for Windows developers.
- *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide*, included online in PDF format, provides information on ODBC for UNIX developers.

For a conceptual overview and programming how-to of ODBC, see:

- Kyle Geiger. *Inside ODBC*. Redmond, WA: Microsoft Press. 1995.

For a review of SQL, see:

- Melton, Jim and Simon, Alan R. *Understanding the New SQL: A Complete Guide*. San Francisco, CA: Morgan Kaufmann Publishers. 1993.
- Groff, James R. / Weinberg, Paul N. *SQL: The Complete Reference, Second Edition.* McGraw-Hill Osborne Media. 2002.

For information about Unicode, see:

- The Unicode Consortium, *The Unicode Standard, Version 5.0*, Addison-Wesley Professional, 2006.
- The Unicode Consortium Home Page at http://www.unicode.org

Conventions used in this guide

TimesTen supports multiple platforms. Unless otherwise indicated, the information in this guide applies to all supported platforms. The term Windows refers to Windows 2000, Windows XP and Windows Server 2003. The term UNIX refers to Solaris, Linux, HP-UX, Tru64 and AIX.

TimesTen documentation uses these typographical conventions:

If you see...	It means...
`code font`	Code examples, filenames, and pathnames. For example, the `.odbc.ini.` or `ttconnect.ini` file.
`italic code font`	A variable in a code example that you must replace. For example: `Driver=`*`install_dir`*`/lib/libtten.sl` Replace *`install_dir`* with the path of your TimesTen installation directory.

TimesTen documentation uses these conventions in command line examples and descriptions:

If you see...	It means...
`fixed width italics`	Variable; must be replaced with an appropriate value.
`[ ]`	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 arguments that you may use more than one argument on a single command line.
`...`	An ellipsis (. . .) after an argument indicates that you may use more than one argument on a single command line.
`%`	The percent sign indicates the UNIX shell prompt.
`#`	The number (or pound) sign indicates the UNIX root prompt.

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

If you see...	It means...
install_dir	The path that represents the directory where the current release of TimesTen is installed.
TTinstance	The instance name for your specific installation of TimesTen. Each installation of TimesTen must be identified at install time with a unique alphanumeric instance name. This name appears in the install path. The instance name “giraffe” is used in examples in this guide.
bits or *bb*	Two digits, either 32 or 64, that represent either the 32-bit or 64-bit operating system.
release or *rr*	Two digits that represent the first two digits of the current TimesTen release number, with or without a dot. For example, 51 or 7.0 represents TimesTen Release 7.0.
jdk_version	Two digits that represent the version number of the major JDK release. Specifically, 14 represent JDK 1.4; 5 represents JDK 5.
timesten	A sample name for the TimesTen instance administrator. You can use any legal user name as the TimesTen administrator. On Windows, the TimesTen instance administrator must be a member of the Administrators group. Each TimesTen instance can have a unique instance administrator name.
DSN	The data source name.

Technical Support

For information about obtaining technical support for TimesTen products, go to the following Web address:

http://www.oracle.com/support/contact.html

1

Tools for Troubleshooting TimesTen

This chapter describes how to use the TimesTen utilities and other tools that are used to diagnose problems with the TimesTen data store. The sections in this chapter are:

Using the ttIsql Utility

The **ttIsql** utility allows you to interactively execute SQL statements and report status information on your data store.

All of the SQL operations described in this guide can be executed from a **ttIsql** `Command>` prompt.

Example 1.1 To start the **ttIsql** utility for the demo data store, enter:

```
% ttIsql demo

Copyright (c) 1996-2006, Oracle.  All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
All commands must end with a semicolon character.

connect "DSN=demo";
Connection successful:
DSN=demo;UID=ankola;DataStore=c:\temp\demo;
DatabaseCharacterSet=US7ASCII;ConnectionCharacterSet=US7ASCII;
DRIVER=C:\WINDOWS\system32\\ttdv
70.dll;Authenticate=0;PermSize=20;TypeMode=0;
(Default setting AutoCommit=1)
```

```
Command>
```

You can then execute SQL statements or **ttIsql** commands at the `Command>` prompt.

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

- **monitor** formats the contents of the SYS.MONITOR table. See "Displaying data store information".
- **dssize** prints data store size information. See "Displaying data store information".
- **showplan** prints the optimizer execution plans for selects/updates/deletes in this transaction. See "Viewing and changing query optimizer plans".
- **isolation** sets / displays the isolation level. See "Working with transactions".
- **timing** prints query timing. See "Timing ODBC function calls".
- **optprofile** prints the current optimizer flag settings and join order. See "Viewing and changing query optimizer plans".

For the full list of **ttIsql** features, see the lists of options and commands under the description of the **ttIsql** utility in the "Utilities" chapter of the *Oracle TimesTen In-Memory Database API Reference Guide*.

Using the ttStatus Utility

You can use the **ttStatus** utility to check the status of the TimesTen daemon and the state of all TimesTen connections.

Example 1.2 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, Subdaemon, or Agent" for troubleshooting information.

On Windows:

```
C:\>ttStatus
ttStatus: Could not connect to TimesTen service: No error
```

On UNIX platforms:

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

Example 1.3 In this example, the output from **ttStatus** indicates the TimesTen daemon is running and it recognizes one data store, named *demo*.

The first line indicates that the TimesTen daemon is running as process 884on port 17000for the TimesTen instance MYINSTANCE. The second line indicates the TimesTen server daemon is running as process 2308 on port 17002.

The output sample indicates the location of the *demo* data store in the file system; that there are currently no connections to the data store; and that the cache agent restart policy for the data store is set to `manual`. (See "Starting and stopping the cache agent" in the *TimesTen Cache Connect to Oracle Guide*.)

Note: This example was produced on Windows. The results are the same on UNIX platforms except for the format of the data store path.

```
C:\>ttStatus

TimesTen status report as of Thu Jan 25 15:45:11 2007

Daemon pid 884 port 17000 instance MYINSTANCE
TimesTen server pid 2308 started on port 17002
TimesTen webserver pid 2188 started on port 17004
------------------------------------------------------------------------
Data store c:\temp\demo
There are 7 connections to the data store
Data store is in shared mode
Shared Memory KEY Global\DBI45b94095.1.SHM.4 HANDLE 0x278

Type            PID     Context     Connection Name              ConnID

Process         4616    0x00d08820  demo                            1
Subdaemon       2136    0x00526768  Worker                       2042
Subdaemon       2136    0x0072e750  Flusher                      2043
Subdaemon       2136    0x007348b8  Checkpoint                   2044
Subdaemon       2136    0x067b0068  Aging                        2045
Subdaemon       2136    0x067c0040  Monitor                      2047
Subdaemon       2136    0x068404c8  HistGC                       2046
Replication policy : Manual
Cache agent policy : Manual
------------------------------------------------------------------------
End of report
```

Example 1.4 In this example, the output from **ttStatus** indicates the TimesTen daemon is running and it recognizes three data stores, named *subscriber1ds*, *masterds*, and *demo*.

The output indicates the *subscriber1ds* and *masterds* data stores each have 2 connections. A closer examination of the output for *subscriber1ds* shows that its located in shared memory segment `DBI45b9471c.2.SHM.2`. The first connection shown in the output identifies the TimesTen subdaemon assigned to manage the *subscriber1ds* data store.

Note: This example was produced on Windows. The results are the same on UNIX platforms except for the format of the data store path and the Shared Memory KEY.

```
C:\>ttstatus
C:\>ttStatus

TimesTen status report as of Thu Jan 25 16:11:34 2007

Daemon pid 5088 port 17000 instance MYINSTANCE
TimesTen server pid 4344 started on port 17002
TimesTen webserver pid 4216 started on port 17004
------------------------------------------------------------------------
Data store c:\temp\subscriber1ds
There are 7 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
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
Cache agent policy : Manual
------------------------------------------------------------------------
Data store c:\temp\masterds
There are 7 connections to the data store
Data store is in shared mode
Shared Memory KEY Global\DBI45b945d0.0.SHM.2 HANDLE 0x270

Type            PID     Context     Connection Name              ConnID
Process         3008    0x00d09008  masterds                          1
Subdaemon       3220    0x00526768  Worker                         2042
Subdaemon       3220    0x0072a758  Flusher                        2043
Subdaemon       3220    0x007308c0  Checkpoint                     2044
Subdaemon       3220    0x00736a28  HistGC                         2046
Subdaemon       3220    0x067f02f8  Aging                          2045
Subdaemon       3220    0x068364a0  Monitor                        2047
Replication policy  : Manual
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 1.5 The *subscriber1ds* and *masterds* data stores are replicated data stores. In this example, the output from **ttStatus** indicates that the replication agents for the replicated data stores have been started. Bidirectional replication has been configured between *masterds* and *subscriber1ds*. There are 5 replication threads created on both data stores by the replication agent.

```
C:\>ttStatus
TimesTen status report as of Thu Jan 25 16:23:33 2007
Daemon pid 5088 port 17000 instance MYINSTANCE
TimesTen server pid 4344 started on port 17002
TimesTen webserver pid 4216 started on port 17004
------------------------------------------------------------------------
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
Cache agent policy : Manual
------------------------------------------------------------------------
End of report
```

Example 1.6 This example shows a connection to an old instance of a data store. This can occur when a data store is invalidated, but some users have not disconnected from the invalidated copy of the data store still in memory. After everyone disconnects, the memory can be freed. This example was produced on a WIndows platform.

```
C:\>ttStatus

TimesTen status report as of Thu Jan 25 16:44:49 2007
Daemon pid 5088 port 17000 instance MYINSTANCE
TimesTen server pid 4344 started on port 17002
TimesTen webserver pid 4216 started on port 17004
------------------------------------------------------------------------
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 ttCapture Utility

The **ttCapture** utility captures information about the configuration and state of your TimesTen system into a file that provides Technical Support with a snapshot

of your system at the time you encountered a problem. When you experience a problem with a TimesTen data store, run the **ttCapture** utility for the data store as soon as possible either when you are encountering the problem or immediately afterward.

The **ttCapture** utility generates a file named `ttcapture.out.<number>`. By default, the file is written to the directory from which you invoke the **ttCapture** utility. You can use the **ttCapture** `-dest` option to direct the output file to be written to another directory.

Note: Directory and file names should always be double-quoted in case there are spaces in them.

On Windows platforms, **ttCapture** also produces a file named `syssum.<number>.nfo` that contains detailed information on your system hardware and configuration.

Example 1.7 On a Windows platform, to capture information related to the data store, *MyDataStore*:

```
% ttCapture MyDataStore
Capturing to file
c:\timesten\tt70\bin\ttcapture.out.20040701.3692
Capturing data store information...
Capturing installation information...
Capturing system information...
Creating msinfo dump in c:\timesten\tt70\bin\syssum.3692.nfo
Finished capture
```

When you contact Technical Support, upload the `ttcapture.out.<number>` generated file to the Service Request. Windows users should also upload the `syssum` file. This can expedite the investigation.

Using the ttDaemonLog Utility

TimesTen uses a *TimesTen daemon* (referred to as the *Oracle TimesTen Data Manager Service* on Windows) and other background processes, known as *subdaemons* and *agents*, to manage access to the data stores. The activities of these TimesTen processes are logged as events in the *TimesTen daemon log*. The daemon log contains both informational messages and error messages. Informational messages are common in the log output, and they are not necessarily an indication that an error has occurred.

TimesTen creates entries in the daemon log as a normal part of its operation. The TimesTen daemon makes a log entry when it starts up and when it shuts down, as well as each time it encounters an error condition. Optionally, you can configure the TimesTen daemon to log each client connect and disconnect. The details for

configuring and using the daemon log are described in "Working with the Oracle TimesTen Data Manager Daemon" in *Oracle TimesTen In-Memory Database Operations Guide*.

Note: Most of information in the daemon log is best interpreted by Technical Support.

Using the ttTraceMon Utility

The **ttTraceMon** utility can be used 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 (level '0' means off) by specifying **ttTraceMon** with the `show` subcommand. The full list of options for **ttTraceMon** is described in the *Oracle TimesTen In-Memory Database API Reference Guide*.

Example 1.8 This example shows that the levels for most all tracing components are set to level 0 (off) for the *demo* data store. (ERR tracing is level 1 by default.)

```
% ttTraceMon demo
LATCH          ... 0
LOCK           ... 0
LOG            ... 0
TRACE          ... 0
API            ... 0
HEAP           ... 0
SM             ... 0
OP             ... 0
EE             ... 0
CG             ... 0
SQL            ... 0
TEST           ... 0
FLOW           ... 0
PT             ... 0
ERR            ... 1
REPL           ... 0
OPT            ... 0
CKPT           ... 0
XA             ... 0
ORACON         ... 0
```

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

Note: TimesTen tracing severely impacts application performance and consumes a great deal of disk space. Only use the **ttTraceMon** utility when

diagnosing problems. When done, reset tracing to the default values shown above.

Starting a trace and reading the trace buffer

You can start a new trace by specifying **ttTraceMon** DSN=*datastore*. For example, to start a trace on the *demo* data store, enter:

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

At the Trace prompt, you can 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 output 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 outputting 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 output the trace buffer 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.

Each line output from the trace buffer has the format:

timestamp sequence component level connection operation

where:

- *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 tracing SQL at level 4, your output will include lines for levels 2 (prepare), 3 (execute), and 4 (open, close, fetch, etc.).
- *connection* is an internal connection number based on the first part of the transaction id (ignore this).

- *operation* 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:

```
16:32:06.328    5281 SQL   2L  3C select cust_num, region, name,
address from xyz.customer
```

Note: With tracing on, the database will be extremely slow. When you are done with your trace on a component, you must turn OFF tracing (level *component* 0).

The remaining sections describe how to do each of the following types of traces:

- SQL tracing
- API tracing
- LOCK tracing
- ERR tracing

SQL tracing

Using **ttTraceMon** with the SQL component provides information about the SQL being prepared or executed by the TimesTen engine. The output levels for SQL tracing is shown below from the least detailed level to the most detailed.

SQL Tracing 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 (e.g., because an index was created), and commands being destroyed. 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 1.9 In this example, we execute **ttTraceMon** to do a SQL trace at level 4 on the *demo* data store. We direct the output from the SQL trace to the *SQLtrace.txt* file.

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

At this point, we execute the application. The trace information is output to the *SQLtrace.txt* file. When the application has completed, we turn off SQL tracing and exit **ttTraceMon**:

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

The contents of *SQLtrace.txt* is shown below:

```
16:32:06.317    5269 SQL   2L  3C Preparing:
16:32:06.317    5270 SQL   2L  3C insert into xyz.customer values (?,?,?,?)
16:32:06.317    5271 SQL   4L  3C sbSqlCmdCompile(E): cmdType:100,
cmdNum:1001130.
16:32:06.328    5272 SQL   3L  3C Executing: insert into xyz.customer values
(?,?,?,?);
16:32:06.328    5273 SQL   3L  3C Executing: insert into xyz.customer values
(?,?,?,?);
16:32:06.328    5274 SQL   3L  3C Executing: insert into xyz.customer values
(?,?,?,?);
16:32:06.328    5275 SQL   3L  3C Executing: insert into xyz.customer values
(?,?,?,?);
16:32:06.328    5276 SQL   2L  3C Preparing:
16:32:06.328    5277 SQL   2L  3C {CALL ttOptUpdateStats('xyz.customer',1)}
16:32:06.328    5278 SQL   4L  3C sbSqlCmdCompile(E): (Found already compiled
version: refCount:01, bucket:05) cmdType:100, cmdNum:1000932.
16:32:06.328    5279 SQL   3L  3C Executing: {CALL
ttOptUpdateStats('xyz.customer',1)};
16:32:06.328    5280 SQL   2L  3C Preparing:
16:32:06.328    5281 SQL   2L  3C select cust_num, region, name, address from
xyz.customer
16:32:06.328    5282 SQL   4L  3C sbSqlCmdCompile(E): cmdType:100,
cmdNum:1001131.
16:32:06.328    5283 SQL   4L  3C Opening: select cust_num, region, name,
address from xyz.customer;
16:32:06.328    5284 SQL   4L  3C Fetching: select cust_num, region, name,
address from xyz.customer;
....
```

```
16:32:06.328    5292 SQL   4L  3C Fetching: select cust_num, region, name,
address from xyz.customer;
16:32:06.328    5293 SQL   4L  3C Closing: select cust_num, region, name,
address from xyz.customer;
```

API tracing

Because many commands are not executed using SQL (for example, transaction commit), it may also be useful to generate API traces. The output levels for API tracing is shown below from the least detailed level to the most detailed

API Tracing Level	Output
1	All rollback attempts by the subdaemon. This will happen 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 data store, as well as commit and rollback for each connection, and a few other operations.
4	+ most everything else conducted at TimesTen's internal API level. It will not show numerous operations on the storage manager and indexes that are done internal to the API.

Note: TimesTen recommends tracing at level 3.

Example1.10 In this example, we execute **ttTraceMon** to do a API trace at level 3 on the *demo* data store. The output from the API trace is sent to the *APItrace.txt* file.

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

At this point, we execute the application. The trace information is output to the *APItrace.txt* file. When the application has completed, we turn off API tracing and exit **ttTraceMon**:

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

The contents of *APItrace.txt* on the normal execution of an application may look something like the following. The format for each line in an API trace is similar to that described the SQL in Example 1.9.

```
15:56:50.165    5031 API      3L    3C sb_dbConnect()(X)
15:56:50.165    5032 API      3L    3C sb_dbConnSetIsoLevel()(E)
15:56:50.175    5033 API      3L    3C sb_xactCommitQ()(E)
15:56:50.175    5034 API      3L    3C sb_xactCommitQ()(E)
15:56:50.175    5035 API      3L    3C sb_xactCommitQ()(E)
15:56:50.175    5036 API      3L    3C sb_xactCommitQ()(E)
15:56:50.175    5037 API      3L    3C sb_xactCommitQ()(E)
15:56:50.175    5038 API      3L    3C sb_xactCommitQ()(E)
15:56:50.175    5039 API      3L    3C sb_xactCommitQ()(E)
15:56:50.175    5040 API      3L    3C sb_dbDisconnect()(E)
```

LOCK tracing

You can 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, will begin to generate a large number of traces, as traces are written for fairly common events. In addition, the traces themselves may be somewhat hard to read in places. If you cannot discern enough information for your purposes, contact Technical Support for more information.

LOCK Tracing 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.

Example1.11 In this example, we execute **ttTraceMon** to do a LOCK trace at level 4 on the *demo* data store and read the results directly from the trace buffer. Before saving the LOCK trace to the buffer, we use the `flush` command to empty the buffer.

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

At this point, we execute the application. The trace information is output to the trace buffer, which is displayed on the screen using the `dump` command:

```
Trace> dump
16:16:42.336   13593 LOCK     4L    3C ENQ: xcb:00003 <DB
0x1312d00,0x0> 0+IX=>
GR activity 0 IX cnt=1
16:16:42.336   13594 LOCK     4L    3C ENQ: xcb:00003 <Cmd
0x155a718,0x0> 0+S=>G
R activity 0 S cnt=1
16:16:42.336   13595 LOCK     4L    3C ENQ: xcb:00003 <Tbl
0x5ff68,0x0> 0+IRC=>G
R activity 0 IRC cnt=1
16:16:42.336   13596 LOCK     4L    3C DEQ: xcb:00003 <Tbl
0x5ff68,0x0> All:N, m
ode=IRC, cnt=1
16:16:42.346   13597 LOCK     4L    3C DEQ: xcb:00003 <Cmd
0x155a718,0x0> All:N,
 mode=S, cnt=1
16:16:42.346   13598 LOCK     4L    3C ENQ: xcb:00003 <Cmd
0x1565964,0x0> 0+S=>G
R activity 0 S cnt=1
16:16:42.346   13599 LOCK     4L    3C ENQ: xcb:00003 <Tbl
0x5ff68,0x0> 0+IRC=>G
R activity 0 IRC cnt=1
16:16:42.346   13600 LOCK     4L    3C DEQ: xcb:00003 <Tbl
0x5ff68,0x0> All:N, m
ode=IRC, cnt=1
16:16:42.346   13601 LOCK     4L    3C DEQ: xcb:00003 <Cmd
0x1565964,0x0> All:N,
 mode=S, cnt=1
16:16:42.346   13602 LOCK     4L    3C ENQ: xcb:00003 <Cmd
0x156a92c,0x0> 0+S=>G
R activity 0 S cnt=1
16:16:42.346   13603 LOCK     4L    3C ENQ: xcb:00003 <Tbl
0x5ff68,0x0> 0+IRC=>G
R activity 0 IRC cnt=1
16:16:42.346   13604 LOCK     4L    3C DEQ: xcb:00003 <Tbl
0x5ff68,0x0> All:N, m
ode=IRC, cnt=1
16:16:42.346   13605 LOCK     4L    3C DEQ: xcb:00003 <Cmd
0x156a92c,0x0> All:N,
 mode=S, cnt=1
16:16:42.346   13606 LOCK     4L    3C DEQ: xcb:00003 <DB
0x1312d00,0x0> All:Y,
```

```
 mode=IX, cnt=1
16:16:42.346   13607 LOCK     4L    3C ENQ: xcb:00003 <DB
0x1312d00,0x0> 0+IX=>
GR activity 0 IX cnt=1
16:16:42.346   13608 LOCK     4L    3C DEQ: xcb:00003 <DB
0x1312d00,0x0> All:Y,
 mode=IX, cnt=1
16 records dumped
```

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

```
Trace > level lock 1
Trace > {press ENTER – blank line}
```

ERR tracing

It may also be useful to trace the ERR component. For example, an ERR trace at level 4 shows all 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.

ERR Tracing Level	Output
1 (set by default)	Fatal errors
4	+ all other error messages, many of which are handled internally by TimesTen.

Example1.12 In this example, we execute **ttTraceMon** to do a ERR trace at level 4 on the *demo* data store. Rather than direct the trace output to a file as in the previous examples, we will read it directly from the trace buffer. Before saving the ERR trace to the buffer, we use the `flush` command to empty the buffer.

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

At this point, we execute the application. The trace information is output to the trace buffer, which is displayed on the screen using the `dump` command:

```
Trace> dump
10:02:30.146    5553 ERR      4L    3C TT3204: Sequence PRODUCTID
is not found -
- file "saCanon.c", lineno 4373, procedure "ppDrSequence"
10:02:30.146    5554 ERR      4L    3C TT3204: Sequence ORDERID
is not found --
file "saCanon.c", lineno 4373, procedure "ppDrSequence"
```

```
10:02:30.146    5555 ERR      4L    3C TT3204: Sequence CUSTID is
not found -- f
ile "saCanon.c", lineno 4373, procedure "ppDrSequence"
10:02:30.146    5556 ERR      4L    3C TT3120: View ORDER_SUMMARY
does not exist
 -- file "eeDDL.c", lineno 3150, procedure "sbEeDrViewEval"
10:02:30.146    5557 ERR      4L    3C TT3204: Sequence ORDERID
is not found --
file "saCanon.c", lineno 4373, procedure "ppDrSequence"
10:02:30.156    5558 ERR      4L    3C TT3204: Sequence PRODUCTID
is not found -
- file "saCanon.c", lineno 4373, procedure "ppDrSequence"
10:02:30.156    5559 ERR      4L    3C TT3204: Sequence CUSTID is
not found -- f
ile "saCanon.c", lineno 4373, procedure "ppDrSequence"
7 records dumped
```

We turn set ERR tracing back to its default setting (1) and exit **ttTraceMon**:

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

Monitoring aging

You can use the **ttTraceMon** utility to verify:

- Aging has started
- How many rows have been deleted by the aging thread
- How many rows remain after the aging thread completes work

Use syntax similar to the following:

```
ttTraceMon(-e level aging n [outfile file]
     {-connStr connection_string | DSN});
```

n represents the trace level. Set *n* as shown in the following table:

Level	Description
1	Display when aging thread wakes up and finds at least one table that is a candidate for aging
2	• Display when aging thread begins acting on a table • Display the summary of rows deleted and remaining in the table when the aging thread has completed work on that table
3	Display aging details, such as rows deleted and remaining, for each table
4	Display when the aging thread wakes up

Higher trace levels display all information displayed by lower trace levels.

Example1.13 `% ttTraceMon -e level aging 4 myDSN`

This command results in output similar to the following:

```
13:33:49.390  322 AGING   4L 2044C    9183P sb_dbAging(E): curTime=156
13:33:50.400  323 AGING   4L 2044C    9183P sb_dbAging(E): curTime=156
13:33:51.410  324 AGING   4L 2044C    9183P sb_dbAging(E): curTime=156
13:33:52.420  325 AGING   4L 2044C    9183P sb_dbAging(E): curTime=156
13:33:52.420  326 AGING   1L 2044C    9183P Entering sbAgingTB(): curTime=157
13:33:52.420  27 AGING   2L 2044C    9183P Entering sbAgingOneTable(): curTime=1
57, ltblid= 638128
13:33:52.420  328 AGING   3L 2044C    9183P curTime=157, numDeleted=0, rows =
2796, tbl = PARENT1
13:33:52.461  329 AGING   3L 2044C    9183P curTime=157, numDeleted=100, rows =
2696, tbl = PARENT1
13:33:53.101  330 AGING   3L 2044C    9183P curTime=157, numDeleted=200, rows =
2597, tbl = PARENT1
13:33:53.161  331 AGING   3L 2044C    9183P curTime=157, numDeleted=300, rows =
2497, tbl = PARENT1
13:33:53.231  332 AGING   3L 2044C    9183P curTime=157, numDeleted=400, rows =
2397, tbl = PARENT1
13:33:53.280  333 AGING   3L 2044C    9183P curTime=157, numDeleted=500, rows =
2297, tbl = PARENT1
13:33:53.340  334 AGING   3L 2044C    9183P curTime=157, numDeleted=600, rows =
2197, tbl = PARENT1
13:33:53.380  335 AGING   3L 2044C    9183P curTime=157, numDeleted=700, rows =
2097, tbl = PARENT1
13:33:53.430  336 AGING   3L 2044C    9183P curTime=157, numDeleted=800, rows =
1998, tbl = PARENT1
13:33:53.481  337 AGING   3L 2044C    9183P curTime=157, numDeleted=900, rows =
1898, tbl = PARENT1
```

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

Using Other Tracing Tools

This section includes the following topic:

ODBC tracing

On Windows, you can use the ODBC trace facility to verify the sequence and content of your commands. Enable tracing by double-clicking **ODBC** in the Control Panel. This opens the ODBC Data Source Administrator. Choose the **Tracing** tab.

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

Using SNMP Traps to Detect Events

Network management software uses SNMP (Simple Network Management Protocol) to query or control the state of network devices such as routers and switches. These devices can generate alerts called *traps* to inform the network management systems of problems.

TimesTen sends SNMP traps for particular critical events to help facilitate user recovery mechanisms. These events are also recorded in the daemon log. Exposing them through SNMP traps allows network management software to take immediate action.

How to configure TimesTen to generate SNMP traps as well as how to receive the traps is described in "Diagnostics through SNMP Traps" in the *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

To understand how network software might be used to detect SNMP traps, you can use the `snmptrapd` program provided in your TimesTen directory: `/install_dir/demo/snmp`. This demo listens on a designated port for SNMP trap messages and either prints the traps to stdout or logs them to syslogd. See the `/install_dir/demo/snmp/README.txt` file for details.

Monitoring the TimesTen System Tables

Each TimesTen data store contains a group of system tables that store metadata about the current state of that data store. The system tables are described in "System and Replication Tables" in the *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

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.

Of particular interest when troubleshooting is the SYS.MONITOR table, which contains statistics about certain events that have occurred since the first connection to the data store. For example, the SYS.MONITOR table describes the number of connections to the data store; the number of checkpoints taken; the size of the data store; the amount of memory currently in use, and so on. You can 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 Example 5.17 in the "Using the ttIsql Utility" chapter of the *Oracle TimesTen In-Memory Database Operations Guide*.

The SYS.PLAN table is useful for troubleshooting performance problems. See "Reading the PLAN table" in the *Oracle TimesTen In-Memory Database Operations Guide for details. You* can check the contents of the SYS.PLAN table by executing SELECT statements on the columns or by using the **ttIsql showplan** command, as described in "Viewing and changing query optimizer plans" section in the "Using the ttIsql Utility" chapter of the *Oracle TimesTen In-Memory Database Operations Guide*.

Using the Query Optimizer

An important tool for performance tuning is the query optimizer.

How to use the query optimizer is described in "The TimesTen Query Optimizer" chapter of the *Oracle TimesTen In-Memory Database Operations Guide* as well as the "Viewing and changing query optimizer plans" section in the "Using the ttIsql Utility" chapter of the *Oracle TimesTen In-Memory Database Operations Guide*.

If you find that a given query runs slower than expected, confirm the query optimizer has the latest statistics for the tables in your query, as described in “Update query optimizer statistics” on page 58. If, after updating your statistics, your query still runs too slow, 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 means of the various **ttOpt** procedures described in "Modifying plan generation" in the *Oracle TimesTen In-Memory Database Operations Guide*.

2

Troubleshooting TimesTen Applications and Data Stores

This chapter helps you diagnose and remedy some of the general problems encountered while using a TimesTen data store.

If you are still having problems with your data store after following the troubleshooting recommendations in this chapter, please contact Technical Support.

The symptoms addressed in this chapter are:

- Unable to Start or Stop TimesTen Daemon
- No Response from TimesTen Daemon, Subdaemon, or Agent
- Application Unable to Create Direct Driver Connection to Data Store
- Application Unable to Create Client/Server Connection to Data Store
- Application Connects/Disconnects are Slow
- Application Disconnects or Exits Unexpectedly
- Transaction is Unexpectedly Rolled Back
- Application is Slow
- Application Unresponsive, Appears Hung
- Application Unable to Find Previously Created Tables
- Running out of a Resource (Memory or Disk)

Unable to Start or Stop TimesTen Daemon

This section describes what to check if you are unable to start or stop the TimesTen main daemon.

Possible cause	What to do
Incorrect privilege	Unless TimesTen was installed for non-root access, you need to have root or ADMIN privileges to start or stop the TimesTen daemon.
Another process using the TimesTen daemon port	Use the **ttStatus** utility to verify the port number assigned to the TimesTen daemon. Confirm another process is not using the same port as the TimesTen daemon. If there is a conflict, either change the port number used by the other process or use **ttmodinstall** to change the port used by TimesTen.
TimesTen daemon is already running	Use the **ttStatus** utility to check if TimesTen daemon is already started. See "Check the current state of TimesTen processes".
Other problems	Use the **ttDaemonLog** utility to check the output of the daemon log for clues.

No Response from TimesTen Daemon, Subdaemon, or Agent

This section describes what to check if one or more of the TimesTen processes appears to be unavailable.

Possible cause	What to do
Daemon, subdaemon, or agent did not start	Check the current state of TimesTen processes
Something caused the daemon, subdaemon, or agent to terminate unexpectedly	Check the TimesTen daemon/event log
If no cause can be detected	Extract a stack trace from the core file

Check the current state of TimesTen processes

You can use the **ttStatus** utility as described in “Using the ttStatus Utility” on page 8 to check the status of the TimesTen daemon and the state of all TimesTen connections.

Example 2.1 If the TimesTen daemon is not running, on Windows you will see:

```
% ttStatus
ttStatus: Could not connect to TimesTen service: No error
```

If the TimesTen daemon has stopped unexpectedly, check the daemon log, as described below. Terminate all application connections to your TimesTen data stores before restarting the daemon, as described in “Working with the Oracle TimesTen Data Manager Daemon” in the *Oracle TimesTen In-Memory Database Operations Guide*.

Note: TimesTen provides a number of SNMP traps to detect the death of TimesTen processes. See "Diagnostics through SNMP Traps" in the *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps* for details.

Check the TimesTen daemon/event log

If you receive an error that indicates the TimesTen subdaemon or agent has stopped, you can look through the TimesTen daemon log, as described in “Using the ttDaemonLog Utility” on page 13, for error messages indicating the reason for its failure.

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

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

Do not restart the daemon until you have terminated all application connections to the data stores. If you terminate the TimesTen daemon/service process, any remaining connections will exit or disconnect and you may invalidate the data stores. The process can be restarted after all connections have exited.

After all applications have exited, you can restart the daemon. See "Working with the Oracle TimesTen Data Manager Daemon" in the *Oracle TimesTen In-Memory Database Operations Guide*. The next connection to each data store will cause TimesTen to recover from the checkpoint and log files.

Extract a stack trace from the core file

If you are experiencing a crash by one of the TimesTen processes on a UNIX system and have exhausted all of the diagnostic options, check to see if TimesTen has generated a core file. If a core file was generated, it can be found at the location of the executable causing the core dump, which in most cases will be the `/var/TimesTen/instance` directory. 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 Technical Support.

On Windows systems you can obtain diagnostic information for a service failure by enabling the 'allow service to interact with desktop' option in the properties dialog for the TimesTen data manager in the **Service** menu. Should a segmentation fault occur in the TimesTen data manager service, you will receive a pop-up that asks if you would like to start the debugger. Contact TimesTen Technical Support and provide the output from the debugger.

Application Unable to Create Direct Driver Connection to Data Store

This section describes what to check if your application is unable to create a direct driver connection to a data store.

Possible cause	What to do
Mismatch between the release of TimesTen and data store	Check release compatibility of data store
Access Control is enabled on the TimesTen data store and user does not have access.	Check Access Control privilege to access to data store
Incorrect file permissions	Check file system permissions to access data store
TimesTen daemon or Data Manager service not running	Check that the TimesTen daemon is running
Incompatible connection attributes or incorrect path name for data store set in the DSN	Check DSN definition
No available shared memory segment or maximum size of shared memory segment too small	Check size and availability of shared memory segments
Not enough swap space	Check available swap space (virtual memory)
Inadequate number of file descriptors	Increase the number of available file descriptors
Other possible causes	Check the TimesTen Daemon Log, as described in "Using the ttDaemonLog Utility"

Check release compatibility of data store

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

See "Data Store Upgrades" in the *Oracle TimesTen In-Memory Database Installation Guide* for details.

Check Access Control privilege to access to data store

If Access Control is enabled on the data store, use **ttUserPrivileges**() to check that you have 'CREATE DATASTORE' privilege. If you do not have access, the administrator must use the GRANT statement to grant you 'CREATE DATASTORE' privilege.

Check file system permissions to access data store

A 'permission denied' error will be generated if you attempt to connect to a data store and you do not have the proper permissions to access the checkpoint or 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** in your DSN.

If the **GroupRestrict** attribute is set for the data store, confirm you are listed in the specified group.

Check that the TimesTen daemon is running

If the TimesTen daemon or Data Manager service is not running, an attempt to connect to a data store will generate TimesTen error 799 (Unable to connect to daemon; check daemon status).

Use the **ttStatus** utility as described in "Check the current state of TimesTen processes" to check the status of the TimesTen daemon.

Check DSN definition

In your DSN description:

- Check DSN attributes
- Check path name to data store and log directories

Check DSN attributes

Certain connection options or DSN attribute settings combinations are not compatible. For example, if **Logging** is disabled, the default row-level lock setting (**LockLevel**=0) cannot be used. In cases where incompatible settings are used, an error will be returned to the application when it attempts to connect to a data store.

Check path name to data store and 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.

On Windows, be careful to distinguish between User and System DSNs in the ODBC Data Source Administrator. Do not create user DSNs because they are

visible only to the user who defines them. System DSNs are visible to all users. In particular, if you run a TimesTen application as a Windows service, it runs as the user "SYSTEM" by default and does not see any User DSNs. Make sure that you are not using a mapped drive in the data store path name.

On UNIX, you must consistently use the same path name to refer to a data store, even if the data store files are reachable through a different path name.

For example, setting the incorrect pathname in the DSN will result in an error such as:

```
% ttIsql -connStr "dsn=demods;DataStore=/incorrect/path"
...
connect "dsn=demods;DataStore=/incorrect/path";
IM002: Data source name not found and no default driver specified
The command failed.The command failed.
```

Check size and availability of shared memory segments

An error is generated if you attempt to connect to or create a shared data store 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 UNIX systems, you can use:

- **ipcs -am** to check if you have other shared memory segments using up memory, such as Oracle instances or other instances of TimesTen.
- **ps -eafl** to see how much memory is being used by running processes.
- **ulimit -a** to see if there are any "per-process" limits on the maximum amount of memory a process can address, maximum file size, maximum number of open files, and so on.

If a shared memory segment is available but too small to hold your data store, 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 data store. The "Changing data store size" section in the *Oracle TimesTen In-Memory Database Operations Guide* describes guidelines for setting the size of your permanent and temporary data partitions. If the amount of memory established for your data store is too large, reset **PermSize** and **TempSize** to smaller values. See "Check the amount of memory allocated to the data store" on page 58 for more information. Another option is to increase the maximum size of the shared memory segment, as described below.

If a data store becomes invalidated because of a system or application failure, a subsequent connection will recover the data store. If recovery fails because you have run out of data store space, then reconnect to the data store with a larger **PermSize** and **TempSize** value than the ones that are currently in effect. If

recovery fails because you do not have enough shared memory, then you should increase the maximum size of the shared memory segment.

To increase the size of the shared memory segment:

- On UNIX system, increase the value of the **shmmax** parameter.
- On Windows systems, increase the amount of available virtual memory, as described in "Check available swap space (virtual memory)" on page 60.

For more information on how to configure the shared memory segment for TimesTen, see "Installation prerequisites" in the *Oracle TimesTen In-Memory Database Installation Guide*.

Check available swap space (virtual memory)

If there is not enough shared memory for the **shmget** operation, then the operation fails. There must be enough swap space to back up shared memory.

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

On Windows systems, you can check and reset the size of your virtual memory from the **Advanced** tab in your **Computer Management Properties** dialog window.

Increase the number of available file descriptors

Each process connected to a TimesTen data store keeps at least one operating-system file descriptor open. Additional file descriptors may be opened for each connection if disk logging is enabled, checkpoints are issued, and transactions are committed or rolled back. If you receive an error that all file descriptors are in use when attempting to connect to a data store, then increase the allowable number of file descriptors.

To increase the number of file descriptors on the system:

- On AIX, the limit is 2048 open files per process so you are not likely to encounter problems with running out of file descriptors.
- On HP-UX, use the System Administration Manager (`/usr/sbin/sam`) to set the value of the **maxfiles** parameter or use the **`ulimit`** (**`limit`** for C-shell users) command (default is 60 open files per process). You can also set the per-process limit programmatically with the **setrlimit** system call.
- On Solaris, use the **ulimit** (**limit** for C-shell users) command (default is 64 open files per process). You can also set the per-process limit programmatically with the **setrlimit** system call.
- On Microsoft Windows systems, the limit is 2000 open files for each process, so you are not likely to run out of file descriptors.

Application Unable to Create Client/Server Connection to Data Store

If your application is unable to create a client/server connection to a data store, first check the procedures described in "Application Unable to Create Direct Driver Connection to Data Store". If these procedures do not address your problem, review the following procedures.

Possible cause	What to do
Incompatible client/server configuration	Check client/server configuration
Access Control is enabled on the TimesTen data store and user does not have access.	Check access control settings
You have not correctly identified the system where the TimesTen Server is running	Check the client DSN
Cannot find TimesTen Server DSN	Check the server DSN
Server failed	Check server log file
TimesTen Server failed to load DRIVER	Verify TimesTen ODBC driver is executable
Application timed out when accessing TimesTen Server	Check the timeout interval
TimesTen Client lost connection with TimesTen Server	• Check the timeout interval • Check network status
Maximum size of shared memory segments too small	Check size and availability of shared memory segments
Failed to attach to shared memory segment for IPC	Check for available file descriptors
Other possible causes	Check the TimesTen Daemon Log, as described in "Using the ttDaemonLog Utility"

Check client/server configuration

Check the compatibility between the supported TimesTen client and server platforms.

Check access control settings

If Access Control is enabled on the data store, check that the attribute **Authenticate**=1 in your DSN and that you have the correct user ID and **PWDCrypt** settings.

Check the client DSN

Check the Server Name Setup values in your client DSN to confirm that you have correctly identified the system where the TimesTen Server is running.

On Windows:

On a Windows client machine, you 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 Data Source Setup dialog.
5. Click **Servers**. This opens the TimesTen Server List.
6. Select the TimesTen Server from the list. This opens the TimesTen Server 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 hostname or network address directly into the Server Name field of the TimesTen Data Source 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. See "Starting and stopping the Oracle TimesTen Data Manager Service on Windows" for information about starting the server manually. See "Testing connections" for more information on identifying this problem.

On UNIX:

On UNIX, specify the TimesTen Server with the **TTC_Server** connection attribute in the ODBC.INI file on the client machine. If the value specified for **TTC_Server** is an actual hostname 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 ServerName, this logical ServerName must be defined in the `/var/TimesTen/sys.ttconnect.ini` file. The TTCONNECT.INI entry for this ServerName needs to correctly define the hostname/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 "Starting and stopping the

daemon on UNIX" for information about starting the server manually. See "Testing connections" for more information on identifying this problem.

Check the server DSN

Check that you have the correct pathname to the data store set in the **DataStore** field in your server DSN. Also confirm that the **Data Source Name** in the server DSN matches the name given in the **TTC_Server_DSN** field in your client DSN. See "Searching for a DSN" for information about the rules TimesTen follows in searching for a DSN.

On UNIX, verify that the Server DSN is defined in the `/var/TimesTen/sys.odbc.ini` file on the machine 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 machine running the TimesTen Server.

If the **GroupRestrict** attribute is set for the data store, confirm you are listed in the specified group. **GroupRestrict** automatically enables the **Authenticate** attribute, so confirm you have provided the correct User Id and password to access the server in your client DSN.

See "Creating and configuring a logical server name" for more information.

Check server log file

The TimesTen server records “connect,” “disconnect” and various warning, error and informational entries in log files. If the TimesTen server fails, check the server's log file. The list of the log entries is available in "List of log entries".

Verify TimesTen ODBC driver is executable

If you are running on a UNIX platform and are unable to load the TimesTen ODBC driver, open the `/var/TimesTen/sys.odbc.ini` file on the machine running the TimesTen Server and locate the Server DSN to which you are trying to connect. Verify that the dynamic library specified in the DRIVER attribute for the Server DSN exists and is executable.

See "Accessing a remote data store on UNIX" and "Creating a DSN on UNIX" for more information.

Check the timeout interval

The default timeout interval for operations is 60 seconds. Long queries may require a longer timeout interval.

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

To set the timeout interval on Windows, see the instructions in "Setting the timeout interval and authentication".

Check network status

Check the TimesTen server log file, as described in “Check server log file” on page 37, 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.

Check for available file descriptors

While using shared memory segments (SHM) as IPC, your 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 6.0 CLIENT] Server is exiting.
Failed to attach to shared memory segment for IPC.
```

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 additional information on file descriptor limits, see "Limits on number of open files".

Application Connects/Disconnects are Slow

This section describes what to check if you encounter slow connects and disconnects to a data store.

Possible cause	What to do
Data store is being recovered	Check if data store is being recovered
ODBC Tracing enabled	Check ODBC Tracing
Other possible causes	Generate an API trace, as described in "Using the ttTraceMon Utility"

Check if data store is being recovered

A slow connect may also indicate that a TimesTen data store 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.

Application Disconnects or Exits Unexpectedly

If an application that is connected to a TimesTen data store abruptly exits one of the following events will occur:

- If there was no outstanding transaction, the connection will be cleanly removed by the TimesTen daemon. Other existing connections will continue processing as if no problem had occurred.
- If there was an outstanding transaction but the application was not in the middle of executing code in the TimesTen library, the transaction will be rolled back and the connection will be cleanly removed by the TimesTen daemon. Other existing connections will continue processing as if no problem had occurred.

This section describes what to check if your application unexpectedly disconnects from the data store or exits.

Possible cause	What to do
Internal application error.	Check for ODBC / JDBC errors
Receipt of an asynchronous signal, such as a keyboard interrupt or an explicit process `kill` command.	Check for ODBC / JDBC errors Check the TimesTen daemon log
Failure of a concurrent application thread.	Check for ODBC / JDBC errors Check the TimesTen daemon log
If using a client/server connection....	• Check the timeout interval • Check network status • Try a direct driver connection
An error in the TimesTen library itself.	Contact TimesTen Customer Support

Check for ODBC / JDBC errors

Check the ODBC errors returned by the **SQLError** function (or JDBC errors using **SQLException**) in all applications to determine whether one of them encountered a problem that caused it to exit prematurely, which in turn may have caused other connections to be forced to disconnect. 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 more extreme cases, 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 21 for details.

Check the TimesTen daemon log

If a TimesTen application exits without returning an ODBC error or any other warning, you can look through the TimesTen daemon log, as described in "Using the ttDaemonLog Utility", for messages indicating the reason for its failure.

Example 2.2 If the application unexpectedly disconnected from the data store, the daemon log might look something like this:

```
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Warn]
daRecovery: examine failed: invalidate (failcode=300002)
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Warn]
1940 ------------------: Invalidating the data store (failcode 300002)
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Info]
1940: daInva
lidateDatabase(c:\temp\demo); shmKey Global\DBI3ee4ba3a.0.SHM.37
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Info]
About to destroy SHM 756
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Info]
shmDestroy:
increment shmSeq to 38, reset panicked and trashed
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Info]
daInvalidate
Database(c:\temp\demo): increment shmSeq to 39, reset panicked and trashed
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Info]
1940: daInvalidateDatabase(c:\temp\demo) done (0)
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Info]
daRecovery/invalidate: clear nUsers, which was 2, set trashed to 39
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]: [Info]
1940: Finished daRecovery for pid 1952.
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[308]: [Warn]
Stopping subdaemon worker for c:\temp\demo because db is invalid.
.....
Oct 01 10:20:13 MYINSTANCE-M700 Oracle TimesTen Data Manager 6.0[1940]:
[Warn]  308 ------------------: subdaemon process exited
```

Try a direct driver connection

If you are encountering unexpected disconnects with client/server connections, try using a direct driver connection, if possible. Doing so will eliminate the network variables and narrow the scope of the problem to either the application or TimesTen.

Transaction is Unexpectedly Rolled Back

Possible cause	What to do
Two connections associated with a single process (Linux only)	Create a “master” connection to the data store

Create a “master” connection to the data store

There is a problem with the Linux process/thread model that may result in two connections being associated with a single process. This can result in an 863 TimesTen error, unexpected rollback of the transaction, or a hung application.

If an application's main process spawns a thread to create the connection, the TimesTen daemon will associate the connection socket with the PID of the spawned thread, rather than the PID of the main process. However the socket will be tied to the main process.

If your application spawns threads that repeatedly connect, disconnect, and exit, there is a potential for the spawned thread that is associated with the connection to exit while the main process is still active. Should the exited thread's PID be assigned to a new process, the TimesTen daemon will then have two socket connections associated with the same PID. The TimesTen daemon cannot differentiate between the two connections and, if one connection exits, the TimesTen daemon assumes the remaining connection is also part of the same process and will attempt to rollback the transaction associated with the remaining connection.

To avoid this problem, rewrite your application to create a “master” connection before creating any threads and keep the master connection open for the life of your application. Any additional connections will be associated with the process for the master connection. You must first stop and restart the TimesTen daemon before executing your updated application.

Application is Slow

See the “Application Tuning” chapter in your TimesTen Developer’s Guide for details on how to maximize the performance of your application and TimesTen data store. This section describes how to detect some of the more common issues that impair performance.

Possible cause	What to do
Inefficient connection type	Check connection type
Outdated database statistics	Update statistics for your tables
Too many table partitions	Check partition counts for the tables
Committing transactions too frequently	Disable autocommit and commit regularly
DurableCommits attribute enabled	Make sure DurableCommit is OFF
Not preparing SQL statements used more than once	Prepare all SQL statements in advance
Wrong indexes or too many indexes	• Create indexes after loading data • Verify your use of indexes
Inefficient use of locks	Verify your isolation and lock levels
Using third-party interface packages	Avoid OLEDB / ADO / third party middleware; use TTClasses instead
Memory leaks	Check for memory leaks in the application
Tracing still enabled for one or more TimesTen components	Check trace settings
Improperly maintained materialized view	Check materialized views
If replication is used, configuration of replication scheme or network environment may be impacting application	See "Poor Replication or XLA Performance"
If Cache Connect is used, Cache Connect configuration or environment may be impacting application	See "Poor Performance of Cache Connect to Oracle"

Check connection type

Client/server connections add significant performance overhead to TimesTen. Driver manager connections can also moderately impact performance. If you need the best possible performance, you should avoid client/server and instead

use a “direct connection” to the TimesTen data store. The performance overhead imposed by client/server connections can be significant, due to the network latencies involved in all communication with the data store.

If your application must run on a different machine from the one hosting the data store, see "Client/Server tuning" for performance-tuning information.

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.

Database statistics are an essential aid to the optimizer for choosing a good 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, you can use the **ttOptUpdateStats** procedure to update the statistics for that table. When updating the statistics for a table, you will 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 will be 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".

Disable autocommit and commit regularly

By default in ODBC, all connections to a data store have autocommit turned on. This means that each individual SQL statement is committed in its own transaction.

In most cases, you can improve the performance of your application by calling the **SQLSetConnectOption** function with SQL_AUTOCOMMIT_OFF to turn off autocommit and explicitly committing multiple SQL statements in a single transaction. This makes particular sense for a large operation, such as a bulk insert or bulk delete.

However, it is also possible to group *too many* operations into a single transaction (thus reducing overall system concurrency, as locks are held for an overly long time). In general, bulk insert/update/delete operations tend to work most effectively when you commit every few thousand rows.

Make sure DurableCommit is OFF

The default setting for **DurableCommits** is OFF. When **DurableCommits** is ON, all transaction log records held in memory are written to disk as part of the "commit" step of a transaction. This is a simple way to guarantee that the transaction is recoverable (barring an actual disk failure).

The default OFF setting for **DurableCommits** improves performance for almost all applications. If **DurableCommits** is ON (**DurableCommits**=1), then every transaction "commit" incurs the cost of a synchronous physical disk I/O. For write-intensive applications this means that TimesTen cannot run at full speed because it is always waiting for the slow disk subsystem. In a worst-case scenario, this can mean that TimesTen performs only marginally faster than a traditional database. When **DurableCommits** is OFF, the physical disk I/O is an asynchronous operation in which batches of updates are written to disk at the same time, thereby preventing the disk from becoming a bottleneck. Making disk I/O an asynchronous operation maximizes performance, but it also means that transactions can be lost.

If the system can tolerate the loss of a small number of transactions, then leave **DurableCommits** set to OFF and use one of the mechanisms below to precisely control your exposure to the possibility of losing transactions. If guaranteed recoverability of every transaction is required, then implement TimesTen replication.

The following mechanisms enable you to precisely control the impact of system failure from the point of view of recoverability:

- Allow the in-memory log buffer to be flushed to disk when it fills. This is the normal behavior of **DurableCommits**=0 (OFF).
- Call **ttDurableCommit** from the application at critical points. This flushes the in-memory log buffer to disk.
- Call **ttDurableCommit** every *n* transactions to flush the in-memory log buffer. For example, call **ttDurableCommit** every 10th or 100th transaction.

If you call **ttDurableCommit** too infrequently, performance may be erratic and durability may be sacrificed. On the other hand, if you call **ttDurableCommit** too often, your application may become I/O bound and performance may suffer. With the right interval, performance is even and fast, and durability is appropriate.

For more information about durable commits, see:

- "When are log files deleted?"
- "Use durable commits appropriately"

Prepare all SQL statements in advance

For best performance, all SQL statements that are executed more than once should be prepared in advance. This is true for all relational databases; but with TimesTen and its extremely fast transaction rates, the time spent to compile a statement can actually take several times longer than it takes to execute it; thus, applications that do not pre-prepare their statements can greatly hinder their TimesTen performance. In addition to preparing statements in advance, input parameters and output columns for those statements should also be bound in advance.

For more information about preparing statements, see "Prepare statements in advance".

Create indexes after loading data

If your application design allows it, you can minimize the time it takes to load data by creating T-tree indexes *after* loading the data. In this situation, the sequence of operations should be:

1. Load data into tables
2. Create T-tree indexes
3. Update statistics
4. Prepare queries

Verify your use of indexes

See "The TimesTen Query Optimizer" for more information.

Knowing how many and which type of indexes to create is important for good database performance. If you create too few indexes, then some frequent database operations will perform more slowly than usual. If you create too many indexes, then insert/update/delete operations will take longer because of the extra time needed to update the indexes. Also, internal deadlocks may occur if a table has too many indexes.

As described in "Indexing Techniques", there are two types of indexes:

- Hash indexes
- T-tree indexes

Hash indexes are automatically created on the primary key of the table and are faster than T-tree indexes for equality (=) searches. For best performance, hash indexes must be sized correctly using the PAGES parameter in the CREATE TABLE statement. An improperly sized hash index degrades the performance of your query.

The formula for sizing the number of pages for a hash index is:

```
expected_number_of_rows / 256
```

Example 2.3 If the *xyz.product* table is to contain 100,000 rows, you can size the hash index on the primary key (*prod_num*) using the formula: 100,000/256 = 390 pages:

```
CREATE TABLE xyz.product (
  prod_num    INTEGER          NOT NULL PRIMARY KEY,
  name        VARCHAR(40)      NOT NULL,
  price       DECIMAL10,2)     NOT NULL,
  ship_weight FLOAT            NOT NULL,
  description VARCHAR(255)     NULL,
  picture     VARBINARY(10240) NULL,
  notes       VARCHAR(1024)    NULL
)
UNIQUE HASH ON(prod_num) PAGES=390;
```

To resize a hash index, use the ALTER TABLE statement with the SET PAGES=CURRENT clause.

Use the CREATE INDEX statement and to create T-tree indexes. T-tree indexes are best for equality and range (>,<,BETWEEN) searches. Unlike hash indexes, T-tree indexes do not need to be sized and can be created and dropped without affecting the table.

Example 2.4 For example, to create a unique T-Tree index, named *xyz.ProdNumIndex*, on the *prod_num* column of the *xyz.product* table, enter:

```
CREATE UNIQUE INDEX xyz.ProdNumIndex
    ON xyz.product (prod_num);
```

There are a few issues to consider when designing your TimesTen table and index schema.

- A well-tuned hash index is faster than the corresponding T-tree index for exact match lookups, but hash indexes cannot be used for range queries (T-tree indexes can be used for both exact match and range lookups, and for sorts, such as for SQL queries involving ORDER BY, GROUP BY, or DISTINCT).
- Full table scan performance over a table with zero T-tree indexes can be improved by including a T-tree index (*any* T-tree index), even if the table scan does not reference columns in that index. This is not intuitive but is easily demonstrated. Thus, you should consider building at least one T-tree index for every table referenced by your application's full table scans even if your application does not have explicit range queries.
- Hash indexes can only be created on the primary key of a table. Thus, there can only be one hash index per table.
- Hash indexes need to be tuned. Use the PAGES= option of the CREATE TABLE statement to specify the number of pages to use for the hash index. Divide the number of rows expected in the table by 256 and use the result for

PAGES=. Specifying too many pages for the index wastes space; specifying too few rows degrades the performance of the hash index because the buckets overflow. You can use the ALTER TABLE statement with the SET PAGES=CURRENT clause to tune the hash index.

- A hash index on three columns can be used only for an exact match lookup of all three columns. Any leading prefix of the columns in a T-tree index can be used for an exact match lookup.

Consider the query:

```
SELECT … FROM t1 WHERE col1 = ? AND col2 = ?
```

Example 2.5 through Example 2.8 show four different index configurations and which index will be selected in each configuration by the optimizer for this query.

Example 2.5 There are two indexes on `t1`:

- Hash index: (`col1`, `col2`)
- T-tree index: (`col1`, `col2`, `col3`)

In this case, both indexes can be used to answer this query.

The hash index can be used because the columns in the WHERE clause exactly match the columns in the hash index; the T-tree index can also be used to answer the query because the columns in the WHERE clause are the leading prefix (the first two) of the columns in the index.

The TimesTen optimizer chooses the hash index because it is faster.

Example 2.6 There are two indexes on `t1`:

- Hash index: (`col1`, `col2`, `col3`)
- T-tree index: (`col1`, `col2`)

In this case, only the T-tree index can be used to answer the query.

The hash index cannot be used because one of the columns in the hash index (`col3`) is not specified in the WHERE clause of the query. The columns of the T-tree index, however, exactly match the columns in the WHERE clause of the query.

The TimesTen optimizer chooses the T-tree index.

Example 2.7 There are two indexes on t1:

- Hash index: (`col1`)
- T-tree index: (`col3`, `col1`, `col2`)

In this case, only the hash index can be used to answer the query.

The hash index can be used because all of its columns are contained in the WHERE clause of the query; note however that there are additional columns in

the WHERE clause of the query, so every row read from the hash index will have the `col2 = ?` predicate applied before the query result is returned.

The TimesTen optimizer chooses the hash index.

Example 2.8 There are two indexes on `t1`:

Hash index: (`col1`, `col2`, `col3`)

T-tree index: (`col3`, `col1`, `col2`)

In this case, neither index can be *efficiently* used to answer the query, although the T-tree index can be used for a full table scan)

The hash index cannot be used for the same reason as in Example 2.6 above. The T-tree index cannot be used because the columns in the query (`col1`, `col2`) are not a leading prefix of the index.

Since neither index can be used, the database must perform a table scan to answer the query (or possibly it will create a temporary index). This degrades performance.

As you can see from these four examples, you have to choose carefully which indexes you create based upon the most frequent queries that you plan to ask the database.

For more information about creating the right indexes, see "Select hash or T-tree indexes appropriately".

Verify your isolation and lock levels

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

An application can acquire locks on the entire data store, 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.

You can check the SYS.MONITOR table or use the **ttXactAdmin** utility, as described in "Check for deadlocks and timeouts" to detect whether an application is spending time waiting for locks. 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 data store. (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 the default Read Committed isolation (**Isolation**=1) level 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.

See "Concurrency control" for more information.

Avoid OLEDB / ADO / third party middleware; use TTClasses instead

Many third-party database interface packages impose a penalty on database performance. These interfaces can be used if performance is not critical in a particular application, but users should be aware of the performance trade-off.

Because these third-party packages can be slow, TimesTen has developed its own C++ ODBC wrapper classes named TTClasses, which is included with TimesTen. For more information on TTClasses, see *Oracle TimesTen In-Memory Database TTClasses Guide*.

Check for memory leaks in the application

TimesTen recommends using Purify for finding the source of memory leaks.

Note: Unless a lot of real memory is being used or leaked, it does not cause a significant slowdown. Slowdowns occur when the RAM usage exceeds the RAM available on the system.

Because the TimesTen library is linked in to your application, running Purify on your application also “purifies” the TimesTen library. There are a small number of expected warnings that Purify will flag in the TimesTen library, which can be suppressed through the use of a suppression file you can request from TimesTen Technical Support.

The suppression file is named `suppress.purify`. To use it, add the following to the PURIFYOPTIONS environment variable:

```
-add-suppression-files='_path_/suppress.purify'
```

To determine if a process is leaking memory, you can use OS utilities to monitor how much memory a process is using. When using such utilities, they typically report the size of each process along with the size of the shared memory segment of the data store. Though it may appear that each process is using a lot of memory, keep in mind that almost all of this memory shared. So the sum of the totals is not meaningful.

When you detect memory leaks in TimesTen, it is often an anomaly in how the “leak” is being measured, rather than a real memory leak. Most operating system utilities (such as “top”) are not useful to identify memory leaks. Because TimesTen uses shared memory segments, additional pages of memory can get touched during runtime.

Purify may detect some memory leaks in TimesTen, but they are all one-time allocations and are not reoccurring. Repeated calls to **SQLAllocEnv** do not result in further allocations, so they should be viewed as fixed size overhead. The TimesTen memory leaks detected by Purify are shown in subsequent sample output. (The size of the leak is the first value; the second value is the number of leaks).

The following leak is allocated once for each thread. As observed, if threads are explicitly created, this thread is freed at thread exit. Solaris turns a non-threaded program into a threaded program having one thread, but Solaris does not properly clean up the single thread at process exit, so this thread is not freed.

```
1064 1 0x22308 vsInitaliseNativeThreadDataForThread <
vsInitaliseThreadDataForThread < vsProcessInitalisation <
vsUnixProcessAttach < SQLAllocEnv <main
```

The following leak is allocated once per process.

```
588 1 0x220a0 vsInitaliseThreadDataForThread <
vsProcessInitalisation < vsUnixProcessAttach < SQLAllocEnv < main
```

The following leak is allocated once per process internally by Solaris in **thr_setspecific()**. The OS doesn't clean it up at process exit. There is no way to prevent this leak.

```
8 1 0x218c0 calloc < _ti_pthread_setspecific <
vsInitaliseThreadDataForThread < vsProcessInitalisation <
vsUnixProcessAttach < SQLAllocEnv < main
```

The following leak is allocated once per process internally by Solaris in **thr_setspecific()**. The OS doesn't clean it up at process exit. There is no way to prevent this leak.

```
1 0x218e0 realloc < _ti_pthread_setspecific <
vsInitaliseThreadDataForThread < vsProcessInitalisation <
vsUnixProcessAttach < SQLAllocEnv < main
```

Check trace settings

Use **ttTraceMon -e show** as described in "Using the ttTraceMon Utility" to confirm tracing is off on all TimesTen components (ERR should be set to 1; all other components should be set to 0).

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.

Check materialized views

Improperly tuned materialized views can greatly impact performance. See "Performance implications of materialized views" and "Improving performance of materialized views" for more information.

Check partition counts for the tables

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

There is a limit of 255 partitions per table. Exceeding this number generates an 8204 error. However, be aware that there is an extra read for each new partition added to a table that slightly degrades performance for a query on the added columns of that table.

The partition value for each table is tracked in the SYS16 column of the system table, SYS.TABLES. You can 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, you can either recreate the table or save and restore the table by using **ttMigrate** create (-c -noRepUpgrade) followed by **ttRestore** (-r -noRepUpgrade). ALTER TABLE DROP does not remove partitions from a table.

Application Unresponsive, Appears Hung

This section describes what to check if your application is unresponsive and appears to be hung.

Possible cause	What to do
Internal application error.	Check for ODBC errors
Lost connection to data store.	Check connection to the data store
Wrong connection attributes set in DSN.	Check connection type
Excessive lock contention.	Check for deadlocks and timeouts
In the middle of a long checkpoint.	Check stack space
Stack over-runs.	Check stack space
Two connections associated with a single process (Linux only)	Create a “master” connection to the data store
Other possible causes.	Check the TimesTen daemon log and gather trace information

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".

If the problem is repeatable, you can use **ttTraceMon** to generate a SQL trace to determine where the application is hanging. See “SQL tracing” on page 16 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 21 for details.

Check connection to the data store

Use the **ttStatus** utility, as described in “Using the ttStatus Utility” on page 8, to check the state of your existing connection to the data store. If your application disconnected, see “Application Disconnects or Exits Unexpectedly” on page 40 for information on how to determine the cause of the unexpected disconnection.

Check for deadlocks and timeouts

If there is no connect problem, a deadlock or timeout may be the problem. The SYS.MONITOR table records information about deadlocks and timeouts. See "Monitoring the TimesTen System Tables" on page 24 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 the number of deadlocks and timeouts are large, consider modifying your application to avoid deadlocks.

If a deadlock occurs, the TimesTen subdaemon eliminates 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, then 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 is unable to 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 will reduce the possibility of lock timeout errors.

System tables are a common source of lock contention. You can reduce contention on the system tables by executing prepared statements, rather than executing the same statements directly each time.

In multi-threaded applications, a thread that issues requests on different connection handles to the same data store may encounter lock conflict with itself. TimesTen resolves these conflicts with lock timeouts.

Check stack space

TimesTen functions share stack space with your application. In multithreaded environments, it is important to avoid overrunning the stack allocated to each thread because consequences can result that are unpredictable and difficult to debug. The amount of stack space consumed by TimesTen calls varies depending on the SQL functionality used. Most applications require a stack space of 16 KB on 32-bit systems and between 34 KB to 72 KB on 64-bit systems.

The amount of stack space allocated per thread is specified by the operating system when threads are created.

On Windows, using the debug driver and linking your application against the Visual C++ debug C library enables “stack probes” that raise an identifiable exception if a thread attempts to grow its stack beyond the amount allocated.

For JDBC applications, you need at least 100KB of native stack space per thread to avoid stack overruns. In addition to the operating system default, the native stack space allocated per thread can also be specified by the Java command line option `-ss` on HP-UX systems or `-Xss` on Solaris systems.

Check the TimesTen daemon log and gather trace information

If you are unable to determine why your application hangs, you can check the daemon log for errors, as described in “Using the ttDaemonLog Utility” on page 13. You can also generate a trace log to detect the activities on various TimesTen components as described in "Using the ttTraceMon Utility". Other, non-TimesTen tools are described in "Using Other Tracing Tools".

Application Unable to Find Previously Created Tables

This section describes what to check if your application is unable to locate previously created tables in the data store.

Possible cause	What to do
No owner or incorrect owner specified	Specify table owner
Access Control is enabled on the TimesTen data store and user does not have SELECT access to tables.	Check Access Control privilege to access tables
Data store is temporary	Check Temporary DSN attribute
Overwrite attribute is enabled	Check Overwrite DSN attribute
Pathname specified in DSN is relative	Check pathname to data store
Lack of data store space	Check available temporary space

Specify table owner

Tables and indexes can be created either with a single name, such as `PARTS`, or with a qualified name incorporating an owner and table name, such as `STAN.PARTS`. When accessing a table or index, if no owner is specified, TimesTen first assumes that the owner is the login ID of the user (the value of **UID**). If TimesTen cannot find the table or index under the user's login ID, it then assumes that the owner is user SYS.

If applications need to connect to a data store as different users and share tables, explicitly specify the owners of the tables when they are created and referenced.

Check Access Control privilege to access tables

If Access Control is enabled on the data store, use **ttUserPrivileges**() to check that you have 'SELECT' privilege for the tables. If you do not have 'SELECT' privilege for the tables, the administrator must use the GRANT statement to grant you the privilege.

Check Temporary DSN attribute

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

Check Overwrite DSN attribute

If the **Overwrite** and **AutoCreate** DSN attributes are enabled and the data store already exists, TimesTen will drop that data store and create a new one. Any tables that were created in the old data store will be dropped.

Check pathname to data store

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

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

rather than:

```
DataStore=demo
```

In the latter case, the data store path name will be relative to the directory where the application was started from. If you are unable to find a table and you are using a relative data store path name, it is possible that the data store in which the table resides in does exist but the data store (checkpoint and log) files are in a different directory than the one that you are accessing.

Check available temporary space

Check the TEMP_ALLOCATED_SIZE, TEMP_IN_USE_SIZE, and TEMP_IN_USE_HIGH_WATER values in the SYS.MONITOR table to confirm you have enough temporary space:

```
SELECT TEMP_ALLOCATED_SIZE, TEMP_IN_USE_SIZE,
       TEMP_IN_USE_HIGH_WATER
       FROM SYS.MONITOR;
```

Note: You can also use the **ttIsql** **dssize** command to list allocated, in-use, and high-water sizes for the temporary data partition.

If you require more temporary space, then reset the **TempSize** attribute in your DSN.

Also, when accessing rows of tables, locks will be acquired and held for a period of time. Keeping transactions short and making sure there is enough temporary space in the data store will prevent 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.

Running out of a Resource (Memory or Disk)

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

Symptom	What to do
Running out of memory space	• Update query optimizer statistics • Check for memory leaks in the application • Check the amount of memory allocated to the data store • Check available swap space (virtual memory) • Compact the data store
Running out of disk space	Check log file use of disk space
Running out of log space	Check log file use of disk space
Running out of file descriptors	Check for available file descriptors
Running out of semaphores	Check the semaphore limit
Running out of CPU	Obtain a stack trace and contact TimesTen Technical Support

Error messages indicate which resources need to be amplified or replenished. Reconfigure the system accordingly.

Update query optimizer statistics

If the data store seems to have enough free space but runs out of data store space when executing a query, make sure you have updated the optimizer statistics with the **ttOptUpdateStats** or **ttOptEstimateStats** procedure. To execute some queries, TimesTen needs to 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 25 for more information on query optimization.

Check the amount of memory allocated to the data store

As described in "Specifying the size of a data store" in the *Oracle TimesTen In-Memory Database Operations Guide*, TimesTen makes use of both permanent and temporary data partitions. The amount of memory allocated for these partitions is set by the **PermSize** and **TempSize** attributes in the DSN definition for the data store.

When the TimesTen data store fills up, it's important to determine whether it is the permanent or the temporary segment that is filling up. You can use the **ttIsql dssize** command to list allocated, in-use, and high-watermark sizes for the permanent and temporary data partitions. The permanent segment consists of table and index data, while the temporary segment consists of internal structures, such as locks, sorting areas, compiled commands, and so on.

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

Permanent segment filling up

If the permanent segment is filling up, compacting the data store, as described in “Compact the data store” on page 60, might help. If the permanent segment remains full, it may mean that the amount of data you have is simply too big. Consider whether you can drop any of the indexes that exist. You may want to look at query plans to see which indexes are actually used. (See "Viewing and changing query optimizer plans" section in the "Using the ttIsql Utility" chapter of the *Oracle TimesTen In-Memory Database Operations Guide* for information on how to generate a query plan.)

You can also use the **ttSize** utility to estimate the amount of memory used by each table in the data store. If the amount of data you need to store is too big, you may need to reset the **PermSize** attribute for the data store to increase the size of the permanent segment. Alternatively, you may need to partition your data into several different data stores if, for example, you cannot shrink the temporary segment or create a bigger data store, due to limits on the memory segment size.

It is sometimes the case that when the permanent segment fills up, copying the data out of the data store, deleting all the data, compacting, and then copying back in will free up space. This can be done more efficiently by using the **ttMigrate** utility with the `-noRepUpgrade` option to migrate the data out, destroy and re-create the data store, and migrate the data back in. This operation is described in "Reducing data store size" in the *Oracle TimesTen In-Memory Database Installation Guide*.

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, as described in "Check available swap space (virtual memory)" below.

Temporary segment filling up

If the temporary segment is filling up, compacting the data store as described above may help. If that doesn't work, it is possible that some commands are allocating too much space due to out-of-date statistics. Try updating the statistics for your tables by calling the **ttOptUpdateStats** or **ttOptEstimateStats** procedure with the invalidate parameter set to ensure the commands are reprepared.

If updating the statistics does not reduce temporary segment memory usage, you can disconnect all connections and then reconnect them. You can verify that all connections have been disconnected by using **ttStatus**. That will free up all temporary space, though you will have to reprepare commands.

If the problem is chronic, you can monitor the data store to try to identify the source of the problem. You can use the **ttWarnOnLowMemory** procedure to enable warnings in the daemon log that indicate that the data store is filling up.

Check available swap space (virtual memory)

If you receive an error indicating that you have run out of swap space, then you may need to increase the amount of available swap space (also referred to as "virtual memory").

On UNIX systems, you can use the **swap** command to check and reset the amount of virtual memory currently established for your system.

On Windows systems, you can check and reset the size of your virtual memory by choosing **Control Panel** > **System** > **Advanced**.

Compact the data store

Before release 6.0, TimesTen applications needed to compact the data store periodically to reduce fragmentation and obtain the most efficient space utilization. Beginning with release 6.0, the internal memory allocation routines achieve best possible space utilization automatically. Explicit compaction is not required.

The mechanics of the internal memory allocation algorithms are complex, particularly in dynamic systems. Transaction semantics and various performance optimizations may result in an incomplete memory picture at a specific point in time. For example, if a database operation acquires 1000 row locks, that space may not be freed immediately after the locks are released. It can be more efficient to reuse the space if a large number of locks will be acquired subsequently. Although TimesTen periodically frees up all unused lock memory and cleans up unused SQL commands, the explicit compaction routines can still be used to trigger accurate point-in-time reporting of memory usage, even though compaction is no longer required to eliminate fragmentation.

You can use the **ttIsql** `compact` command or the **ttCompact** procedure to force cleanup and compaction of all currently freeable memory. Memory usage statistics collected may then present a more accurate picture.

It is important to note that the compaction operation (implicit or explicit) does not relocate any items that are in use. So if two adjacent pieces of memory are only half-full, it will not move the in-use items from one onto the other, so that the other can be freed. To free up such space, the **ttMigrate** utility with the `noRepUpgrade` option can be used to save and restore the data store, as described

in "Reducing data store size" in *Oracle TimesTen In-Memory Database Installation Guide*.

Check log file use of disk space

As described in "Checkpointing" in the *Oracle TimesTen In-Memory Database Introduction*, TimesTen saves a copy of the data store in one of two checkpoint files, which are stored in the directory specified by the **DataStore** attribute. Each checkpoint file can grow on disk to be equivalent to the size of the data store in shared memory. For each permanent data store, you must have enough disk space for the two checkpoint files and for log files (assuming logging to disk is enabled).

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, log files accumulate in the directory specified by the **DataStore** attribute. The maximum size of your log files is set by the **LogFileSize** attribute.

When a disk fills up with TimesTen data, it is most often due to a build-up of log files. Log files are used for numerous purposes in TimesTen, including checkpointing, backups, and replication. It is important to determine which operation is putting a “hold” on the log files, so that appropriate action can be taken to allow the 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.

- **Checkpoint** — If a TimesTen application crashes and the data store needs to be recovered, the checkpoint and log files are used to recover the data. The “most recent” log files are used -- those written since the checkpoint was done. So log files accumulate during the interval between checkpoints. Your application should periodically call the **ttCkpt** or **ttCkptBlocking** procedure to checkpoint the data and free up the space on the disk. If checkpoints are done very infrequently, a large number of log files may accumulate, particularly if many changes are made to the data store during that interval. See"Checkpointing" for an overview of checkpointing in TimesTen.
- **Replication** — TimesTen replication transmits changes to one data store to one or more other data stores. It does this by reading the log and sending over any relevant changes. If replication is paused, the log files will build up. To prevent log build-up, avoid pausing replication for too long. Delete subscriptions entirely, and reset replication where appropriate. See "Setting the replication state of subscribers" for more information on pausing and restarting or resetting replication.
- **Backup** — TimesTen supports an incremental backup facility that will augment a backup with changes made since the last backup. It uses log files to augment the backup. So log files accumulate during the interval between incremental backups. To avoid a large log build-up, make sure to do

incremental backups at relatively frequent intervals. Or, if desired, disable incremental backups and do full backups instead. See "Copying, migrating, backing up and restoring a data store" for more information on backups.

- **XLA** — TimesTen's persistent XLA facility reports changes to the data store. It uses log files to do this. Log files are kept around for use by XLA until the corresponding transactions are acknowledged using the **ttXlaAcknowledge** function. If this routine is called infrequently, log files will build up, so it's important to call it periodically. See "Retrieving update records from the transaction log" for more information on acknowledging XLA records.
- **XA** — TimesTen's XA support makes use of log files to resolve distributed transactions. If these transactions are not resolved in a timely manner, log files will build up. See "Distributed Transaction Processing XA" for more information on XA.
- **Long-Running Transactions** — TimesTen uses the transaction log to roll back transactions. So a log hold is placed for the duration of a transaction. Transactions that are active for a long time result in a log-file build-up if the transaction has written at least one log record (that is, it is not a read-only transaction). So it is important to commit write transactions with reasonable frequency to avoid significant log-file build-up. See "Size transactions appropriately" for more information on transaction length.

  If an application has transactions that delete many rows, you may need to free extra space on the disk where your temporary directory resides. TimesTen creates temporary files when large amounts of space in a data store are freed by a transaction. See "Changing data store size" for details on changing the size of the data store's temporary data partition. You can also change the location of your temporary directory by setting the `TMP` environment variable on Windows or the `TMPDIR` environment variable on UNIX, as described in "Problems using ttRepAdmin -duplicate".

Two TimesTen attributes are related to disk use:

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

Check the semaphore limit

When creating multiple client/server connections to a TimesTen data store configured to allow shared memory segment as IPC, you may encounter errors that indicate TimesTen was unable to create a semaphore.

On Solaris, you can view the current semaphore limit, with:

```
% sysconfig -q ipc
```

If you require more than 6 ShmIpc-enabled Client DSN connections per process, you need to increase the number of semaphores as described in "Increase number of semaphores" in the *Oracle TimesTen In-Memory Database Installation Guide*.

3

Troubleshooting Cache Connect to Oracle

This chapter describes how to troubleshoot some of the problems you may encounter when using Cache Connect to Oracle. It includes the following topics:

Unable to Start or Stop the Cache Agent

Possible cause	What to do
Cache agent already running	See "Check status of the cache agent".
Incorrect Cache Connect to Oracle settings	See "Check status of the cache agent" and "Check ORACLE_HOME environment variable".
Unable to locate Oracle libraries	See "Check LD_LIBRARY_PATH environment variable".
Wrong version of Oracle client libraries	32-bit TimesTen must use 32-bit Oracle client libraries. 64-bit timesTen must use 64-bit Oracle client libraries.
Access Control is enabled on the TimesTen data store.	If Access Control is enabled on the data store, you must have ADMIN privileges to start or stop the cache agent.

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" to check the status of the cache agent.

If the cache agent is not running, then start it as described in "Starting and stopping the cache agent" in the *TimesTen Cache Connect to Oracle Guide*. If attempts to start the cache agent fail, then investigate the possible causes and reboot the machine before attempting to start the cache agent.

Check the Oracle Database version

Check that the correct version of the Oracle Database is installed on the machine.

If you are on an AIX machine, check that:

- The link /`opt/TimesTen/tt60/lib/libttor.so` points to:
  - `/opt/TimesTen/tt60/lib/libttor_9i.so` for Oracle9i
  - `/opt/TimesTen/tt60/lib/libttor_10g.so` for Oracle Database 10g
- The link `/opt/TimesTen/tt60/bin/timestenorad` points to:
  - `/opt/TImesTen/tt60/bin/timestenorad_9i` for Oracle9i
  - `/opt/TimesTen/tt60/bin/timestenorad_10g` for Oracle Database 10g

Check ORACLE_HOME environment variable

On UNIX platforms, 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 "Setting up TimesTen and Oracle" in the *TimesTen Cache Connect to Oracle Guide*.

Check LD_LIBRARY_PATH environment variable

Make sure the LD_LIBRARY_PATH environment variable is set for both your application and TimesTen daemon environments and that it includes the path to the TimesTen libraries and Oracle libraries, as described in "Shared library path environment variable".

On most UNIX platforms: Use the **ldd** command to verify that the cache agent can locate the Oracle library (`timestenorad`). The LD_LIBRARY_PATH environment variable is set in the TimesTen daemon startup script. You must restart the main daemon after changing the LD_LIBRARY_PATH in the script.

On HP_UX: Use the **ldd** command to verify that the cache agent can locate the Oracle library (`timestenorad`). The SHLIB_PATH environment variable is set in the TimesTen daemon startup script. You must restart the main daemon after changing the SHLIB_PATH in the script.

On Windows: Verify that the TimesTen (`tt*.dll`) libraries are in your Windows 'system32' folder and that the pathname to the 'system32' folder is included in your system PATH environment variable (for example: `C:\winnt\system32`).

See "Install the Oracle Client on the TimesTen host" for more information on how to set up TimesTen to operate with Oracle.

Unable to Resolve Oracle Service Name

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

- Use the Oracle TNSPING utility to verify that the service can be reached.
- Ensure that the **OracleID** set in your DSN definition matches the Oracle Service Name for the Oracle 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 server and confirm that there is a service name or a SID that identifies the Oracle server. If you add or modify a service name, you may need to reboot.

  Check the cache agent user name and password on Oracle with SQLPlus to make sure this service name works. For example,

  ```
  %sqlplus scott/tiger@OracleHost
  ```

  where *scott* is the cache agent ID, *tiger* is the cache agent PWD, and *OracleHost* is the **OracleID** specified in your DSN definition.

Note: Your cache agent account may be different from your regular Oracle user account. See "Create Oracle users and set privileges" for details.

- Ensure that there is only one copy of `tnsnames.ora` on your TimesTen machine. Also check the permission on `tnsnames.ora`.
- If you are running TimesTen on a UNIX system, check that the ORACLE_HOME environment variable points to your *Oracle_installation* directory.

  For example, `ORACLE_HOME=/products/oracle10g`
- If you are using an Oracle 9.0.x client on Windows, you may discover that the cache agent cannot to start even when the **OracleID** connection attribute is set correctly. To verify the **OracleID** is set correctly, use **ttIsql** to issue a query to Oracle using **PassThrough**=3 connection Setting. For example:

  ```
  ttIsql
  Command> connect
  "dsn=myDSN;uid=scott;pwd=tiger;passthrough=3";
  Command> SELECT * FROM DUAL;

  1 row found.
  ```

  If the connection succeeds, your **OracleID** is set correctly and the problem is that you are using a 9.0.x version of the Oracle client. This problem can be resolved by upgrading to a Oracle 9.2.0.8.0 client.

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 data store.

This can occur when you are trying to use Cache Connect and Oracle on the same machine and the TNS_ADMIN environment variable does not point to the proper TNSNAMES.ORA file for Oracle. 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 running on different machines. In this case, do not reset the TNS_ADMIN environment variable to point to a TNSNAMES.ORA file on the machine where TimesTen is running. The Oracle 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 will be unaware of the TNS_ADMIN setting. Cache Connect cannot operate properly when the Oracle 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 wish to use. You can include other TNSNAMES.ORA files with the INAME command inside the TNSNAMES.ORA file.

Unable to Validate Oracle Username and Password

If you receive an error that indicates "Validation of the Oracle usernames and passwords failed" and a daemon message that indicates "OCIInitialize failed. Return code -1," the problem is probably due to Cache Connect to Oracle not initializing the OCI library.

Possible cause	What to do
The library environment variable is not set correctly	Check library path environment variable.
Oracle client processes not running	Check status of TNS Listener and Oracle server.
User does not have the correct Oracle privileges	Check Oracle privileges.
Incorrectly configured DSN	Check DSN definition.
TimesTen daemon still running in pre-Cache Connect to Oracle environment	Reboot TimesTen machine.
Problems with cache administration user ID or password	Set the cache administration user ID and password.
System cannot locate `ORACLE_HOME/bin` directory	Confirm system environment can locate ORACLE_HOME/bin.
Inconsistent user and system environments	Check user and system environment.
Dynamic libraries not loading	Verify the loaded dynamic libraries.

Check library path environment variable

Check the library path environment variable:

On this platform...	Check this variable...
UNIX except HP-UX	LD_LIBRARY_PATH On 64-bit platforms, LD_LIBRARY_PATH64 takes precedence over LD_LIBRARY_PATH. Make sure that the library path is specified in LD_LIBRARY_PATH64.
HP-UX	SHLIB_PATH
Windows	PATH

The library path environment variable must include the following:

TimesTen and Platform Bit Combination	Setting
64-bit TimesTen or 32-bit TimesTen on 32-bit platform	$ORACLE_HOME/LIB and $ORACLE_HOME/NETWORK/LIB
32-bit TimesTen on 64-bit platform	$ORACLE_HOME/LIB32 and $ORACLE_HOME/NETWORK/LIB32

Check status of TNS Listener and Oracle server

On Windows, check the Oracle services under Control Panel > Administrative Tools> Services and confirm the Oracle client services are running.

On UNIX, use the **ps** command to confirm the Oracle client processes are running.

Check Oracle privileges

From an Oracle SQL*Plus command prompt, list the current Oracle 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 Cache Connect to Oracle operations that are specified in "Create Oracle users and set privileges". Contact your Oracle Administrator if you require additional privileges.

Check DSN definition

- Confirm you have correctly set the DSN attributes as described in "Defining DSNs for cached tables".
- Confirm that the DSN definition for Cache Connect to Oracle is a *system* DSN.
- Confirm that the DSN for Cache Connect to Oracle is defined only once.
- Confirm Oracle user name and password. Use SQLPlus and connect to Oracle using the same **Oracle User ID** and **OraclePWD** used in your DSN definition to confirm they are correct.

Reboot TimesTen machine

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

Set the cache administration user ID and password

From a ttIsql session, enter the following:

```
Command> call ttCacheUidPwdSet('scott','tiger');
```

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

Confirm system environment can locate ORACLE_HOME/bin

Make sure the system environment can locate the Oracle installation directory, `ORACLE_HOME/bin`.

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 UNIX).

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

   ```
   % install_dir/srv/ttsrv70.exe -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 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 pathname to the `oci.dll` library between the user and system environments?
6. If you detect differences, make the necessary modifications.
7. Reboot the system and restart the TimesTen daemon.

Verify the loaded dynamic libraries

If you are running on a Windows system with Virtual C++ installed, you can 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 `ttora70.exe` and highlight it. Right-click on it and select Debug. This will bring you into Virtual C++ and you should see the loaded dll in the debug window, as described in "Check LD_LIBRARY_PATH environment variable".
4. Load the cache group to force an cache connection from the cache agent:

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

5. Compare the loaded dll in your debug window with the partial list shown in Example 3.1.

Example 3.1 This partial list was created with the Oracle 10.2.0.1.0 client.

```
Loaded symbols for 'C:\tt70\bin\ttora70.exe'
Loaded 'C:\WINDOWS\system32\ntdll.dll', no matching symbolic
information found.
Loaded 'C:\WINDOWS\system32\kernel32.dll', no matching symbolic
information found.
Loaded symbols for 'C:\tt70\bin\tten70d.dll'
Loaded symbols for 'C:\tt70\bin\ttco70d.dll'
Loaded 'C:\WINDOWS\system32\wsock32.dll', no matching symbolic
information found.
Loaded 'C:\WINDOWS\system32\ws2_32.dll', no matching symbolic
information found.
Loaded 'C:\WINDOWS\system32\msvcrt.dll', no matching symbolic
information found.
```

```
Loaded 'C:\WINDOWS\system32\ws2help.dll', no matching symbolic
information found.
Loaded 'C:\WINDOWS\system32\advapi32.dll', no matching symbolic
information found.
...
```

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 Oracle 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 for Cache Connect to Oracle" in *TimesTen Cache Connect to Oracle Guide*.

NULL Constraint Does Not Match Oracle

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 Oracle 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 Oracle does not have a NULL constraint, but col2 in the cache group is defined as NOT NULL.

Unable to Create a Cache Group

This section describes some of the errors and warnings you might encounter when executing the CREATE CACHE GROUP statement.

Possible cause	What to do
User does not have the correct Oracle privileges to create the cache group type.	Check Oracle privileges.
Access Control is enabled on the TimesTen data store and user has insufficient access to data store.	If Access Control is enabled on the data store, you must have DDL privileges to create a cache group.
Access Control is enabled on the TimesTen data store and the internal/external user does not match the Oracle user.	If Access Control is enabled on the data store, the TimesTen user name must be the same as the Oracle user name.
Cannot connect to Oracle	See: • "Unable to Resolve Oracle Service Name" • "Unable to Resolve Connect Identifier" • "Unable to Validate Oracle Username and Password" Check whether Oracle needs to be restarted. Check the network status.
Cache administration user ID or password not set (when trying to create AWT or AUTOREFRESH cache groups)	Set the cache administration user ID and password.
Unsupported data type mapping	See "Unsupported Data Type Mapping".
Different nullability setting in Oracle	See "NULL Constraint Does Not Match Oracle".

Autorefresh Not Refreshing Cache at the Specified Interval

To verify the health of autorefresh, you can periodically call the **ttCacheMonitor** procedure from **ttIsql**. There are two **ttCacheMonitor** features that indicate the progress of autorefresh:

- cgRefreshCount
- cgTotalNumRows

Use the `cgRefreshCount` feature to check the current value of an internal counter associated with your cache group. The counter is reset to 0 when the cache agent is restarted and is incremented after each autorefresh on the cache group. If

autorefresh is healthy, you should see the output from `cgRefreshCount` increasing over time.

You can also use the `cgTotalNumRows` feature to check the total number of root table rows that get refreshed since the cache agent was started. The output from `cgTotalNumRows` increases only when there are changes in the Oracle base table.

For example, to use **ttCacheMonitor** check the number of autorefreshes that have occurred on the cache group, *cg1*, since the cache agent was started, enter:

```
% ttIsql
Command> call ttCacheMonitor('cgRefreshCount', 'cg1', null);
< 11 >
1 row found.
```

To check the total number of rows in the root table that were updated by the refreshes, enter:

```
Command> call ttCacheMonitor('cgTotalNumRows', 'cg1', null);
< 1 >
1 row found.
```

If you discover that your cache group is not automatically refreshing with Oracle updates every specified interval, check the possible causes in the table below. Most problems related to autorefresh are reported in the daemon log generated by the **ttDaemonLog** utility.

You can enable SNMP traps to alert you when autorefresh problems occur and point to the daemon log for details. The SNMP traps related to Cache Connect to Oracle autorefresh are:

- **ttCacheAutoRefQueFullTrap**
- **ttCacheIncAutoRefFailedTrap**
- **ttCacheValidationErrorTrap**
- **ttCacheValidationWarnTrap**
- **ttCacheValidationAbortedTrap**

The following table shows possible causes for autorefresh problems.

Possible cause	What to do
Cache agent not started with a cache administration user	Specify a cache administration user ID and password when starting the cache agent, as described in “Starting and stopping the cache agent”.
Object ID of the base table has changed.	Recover and reset autorefresh Oracle objects.
Autorefresh trigger not enabled	Recover and reset autorefresh Oracle objects.

Possible cause	What to do
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.	Recover and reset autorefresh Oracle objects.
There is no row in the TT_*version*_USER_COUNT table with usercount > 0 for every active incrementally autorefresh table	Recover and reset autorefresh Oracle objects.
Log table is empty	Recover and reset autorefresh Oracle objects.
User count is less than 0 or any TT_*version*_USER_COUNT log sequence anomalies	Recover and reset autorefresh Oracle objects.
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 "Recover and reset autorefresh Oracle objects" procedure. Check the daemon log for messages about "fatal anomalies". This indicates corrupt or missing Oracle 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 "Recover and reset autorefresh Oracle objects" procedure. Check the daemon log for messages about "fatal anomalies". This indicates corrupt or missing Oracle objects.
If the current log sequence number in the TT_*version*_USER_COUNT table changes, is different from the bookmark and the associated cached table is not refreshed by the next committed autorefresh.	First, try restarting the cache agent. If that does not work, follow the "Recover and reset autorefresh Oracle objects" procedure.
Resource problem	Restart the cache agent.

Reset autorefresh state

Incremental AUTOREFRESH will not work if the TRUNCATE statement is used on an Oracle base table. If TRUNCATE is used on an Oracle 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 Oracle objects

If you know or suspect the Oracle objects used by AUTOREFRESH is the cause of the problem, you can use the following procedure to re-create the Oracle objects.

1. Use ALTER CACHE GROUP to reset the AUTOREFRESH state to OFF on all cache groups on all data stores 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 data stores.

3. Check if the user count is zero for each table in the cache group.

   On the Oracle database, execute:

   ```
   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 will perform a clean up operation. It will display the following message to the daemon 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 <cachegroup 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 data stores.

Incremental Autorefresh Not Progressing

If incremental autorefresh is not progressing, verify that:

- Autorefresh state is ON
- Cache agent is running

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

Condition	What To Do
Oracle server connection errors or warnings	See "Application Unable to Create Client/Server Connection to Data Store" 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.
Insufficient permanent data partition errors on TimesTen	Increase **PermSize**.
Autorefresh Oracle object validations errors or warnings	See "Recover and reset autorefresh Oracle objects".
Cache agent exits unexpectedly.	Contact Technical Support.
Core files in main daemon directory	Contact Technical Support.
Warnings about incremental autorefresh becoming full refresh	See "Incremental Autorefresh Becomes Full Autorefresh".
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. **Note**: Cache groups with the same autorefresh interval are autorefreshed in one transaction.

Validate autorefresh Oracle objects

The cache agent automatically verifies that Oracle 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 daemon log. If you see object validation errors, contact Technical Support unless one of the following conditions has occurred:

- The TimesTen data store 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.

The cache group needs to be re-created if one of the preceding conditions has occurred.

Incremental Autorefresh Becomes Full Autorefresh

If you are using incremental autorefresh, you may see messages in the daemon log that indicate full autorefresh operations are taking place, such as:

```
10:19:00 Info: 10290:14 ---: Performing a full refresh with root: tablename
```

This usually occurs in one of the following situations:

- The autorefresh state is turned ON before the cache group has been loaded.
- The cache administration user tablespace becomes full.

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 table, change log tables for each Oracle 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 system tablespace.

You can specify the tablespace when you create the cache administration user on Oracle. You can also specify the tablespace after user creation with the Oracle ALTER USER statement.

Change log tables for each of the cached Oracle tables reside in the cache administration user tablespace. For each update on an Oracle table, one row (a change log record) is inserted into the change log table for that Oracle 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 table and on the autorefresh interval on TimesTen. TimesTen removes the change log records every 20 seconds that have been autorefreshed by all caches.

When the cache administration user tablespace gets full, the autorefresh trigger makes space for new change log records by deleting existing change log records. This can cause an automatic full refresh for some tables on some TimesTen data stores. This degrades performance.

If you are using AUTOREFRESH in INCREMENTAL mode and suspect your AUTOREFRESH queries are overflowing the change log table, use the **ttDaemonLog** utility as described in "Using the ttDaemonLog Utility" to determine the frequency of FULL AUTOREFRESH operations.

Check for the following conditions if the tablespace is full:

- Is a cache group being created or is a data store being duplicated? Both of these operations temporarily stop clean-up operations on the change log table.
- Are the cache agents on all TimesTen data stores running?

You can decrease the AUTOREFRESH INTERVAL value or increase the size of the change log table by specifying a WITH LIMIT value in the AUTOREFRESH INCREMENTAL clause.

Poor Performance of Cache Connect to Oracle

Possible cause	What to do
Inefficient AUTOREFRESH queries	Check AUTOREFRESH setting
Change log table is too small	See "Incremental Autorefresh Becomes Full Autorefresh".

Check AUTOREFRESH setting

The efficiency of AUTOREFRESH operations can impact Cache Connect to Oracle performance. Factors that might impact AUTOREFRESH performance include:

- INTERVAL value
- Log limit
- Number of cache groups with the same INTERVAL value
- Depth of cache group (how many generations of descendants)
- Number of rows in the cached tables
- Rate of changes to the tables
- Size of a row
- Tablespace size

Problems with Cache Administrator

This section describes some of the possible problems you may encounter with Cache Administrator.

Possible cause	What to do
Web server not running	Check web server.
Cache Administrator cannot access the columns list	Check the type of DSN defined for your data store.

Possible cause	What to do
“Page Cannot Be Displayed” error when opening the Cache Administrator	Check URL and web server configuration.
The Cache Administrator does not allow login	Check Cache Connect to Oracle attributes in the DSN.
Selected child tables do not appear in the cache group hierarchy and the error “child tables are not yet added” is displayed	Define table hierarchy.

Check web server

Use the **ttStatus** utility as described in “Using the ttStatus Utility” to check the status of the web server process. For example, if the web server is running **ttStatus** will report the web server as ‘started’, along with the process id and port number:

```
TimesTen status report as of Tue Oct 05 13:45:31 2005

Daemon pid 556 port 16000 instance tt60
TimesTen server pid 1168 started on port 15102
TimesTen webserver pid 1108 started on port 15104
```

If the web server is not running, you can use the **ttDaemonAdmin** utility to start it:

```
% ttDaemonAdmin -startwebserver
```

Check the type of DSN defined for your data store

When using the Cache Administrator to access a columns list from the Oracle database, you must define your DSN as a System DSN. The columns list cannot be accessed from the Cache Administrator if the DSN is a User DSN. See “Data source names”.

Check URL and web server configuration

Check the following:

- Has the web server been configured correctly? See the *Oracle TimesTen In-Memory Database Installation Guide*.
- Are you using the correct URL? It should be http://*machine_name*:*port*/cache. *machine_name* is the machine where the TimesTen daemon is running, and *port* is the port number of the daemon's Web server. The port number was specified during installation. If you do not know the port number, look in the daemon's Web server log, `webserver.log`, in the daemon's directory.

- Are you using the correct host and port number. You can use the **ttStatus** utility to check the port number and the **ttmodinstall** utility to change the port number, if necessary.
- Is Cache Administrator on your local machine? By default, the Cache Administrator can only be accessed locally on the TimesTen host. You can edit the HOSTSALLOW and HOSTSDENY parameters in the `webserver.config` file to grant or deny access to the Cache Administrator. from other hosts.

Check Cache Connect to Oracle attributes in the DSN

Check the following:

- Does the DSN exist?
- Is the DSN string syntax correct?
- Is the Oracle ID set in the DSN string?
- Is the Oracle ID valid? Through SQL*Plus, can you connect to the Oracle instance using the Oracle ID?
- Are the user name and password correct?

Make sure that the perl that is in use (derived from the $PATH environment variable) is the same as the installed perl. The location of the installed perl can be found in the PERL and PERLLIB parameters of the `webserver.config` file.

If you are running on Windows, make sure your PATH environment variable includes the path to your Windows 'system32' folder (for example: `C:\winnt\system32`)

Define table hierarchy

After selecting the root and **other tables** on the **Create a Cache Group Definition** page, the Cache Administrator does not display the other tables within the cache hierarchy until they have been added. For example, if there are three tables in the cache group: ROOT, TABLE1 and TABLE2, there are three possible hierarchies for the cache group:

- TABLE1 and TABLE2 are children of ROOT.
- TABLE1 is a child table of ROOT and TABLE2 is a child table of TABLE1.
- TABLE2 is a child table of ROOT and TABLE1 is a child table of TABLE2.

As the number of tables increases, so does the number of possible hierarchies. The Cache Administrator does not compute the various combinations. You must explicitly define the group hierarchy from the list of selected tables.

Problems with AWT Cache Groups

Creating an asynchronous writethrough (AWT) cache group automatically creates a replication scheme that allows the data store to communicate with the Oracle database. You must start the replication agent after you create an AWT cache group and start the cache agent. See "Setting up an AWT cache group" in *TimesTen Cache Connect to Oracle Guide*.

This section summarizes material in Chapter 4, "Troubleshooting Replication" that is useful for troubleshooting AWT cache group problems. It includes the following topics:

- Unable to start or stop replication agent
- Replication does not work
- Poor AWT performance
- Using SNMP traps for notification of replication events

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
Access Control is enabled and you do not have ADMIN privileges	If Access Control is enabled on the data store, you must have root or ADMIN privileges to use the **ttAdmin** utility or the **ttRepStart** or **ttRepStop** procedures to start or stop a replication agent.
TimesTen daemon not started	Check the state of the TimesTen daemon, as described in "Check the current state of TimesTen processes". If necessary, start the TimesTen daemon as described in "Working with the Oracle TimesTen Data Manager Daemon".

Replication does not work

If you are unable to get replication working, the problem may be one or more of the following:

Possible cause	What to do
TimesTen daemon and/or replication agents not running	Check status of TimesTen daemon and replication agents.
Replication agents not communicating	Check that replication agents are communicating.
Replication not in **Start** state	Check replication state.

Poor AWT performance

This section addresses issues that may impact AWT performance.

Possible cause	What to do
Slow network	Check network bandwidth.
Log buffer too small	Check size of log buffer.
Frequent or inefficient disk writes	Check durability settings.
Reading from log files on disk rather than the log buffer	Check for reads from log files.

Using SNMP traps for notification of replication events

TimesTen can send SNMP traps for certain replication events to enable network management software to take immediate action. The possible SNMP traps that can be sent by TimesTen are:

- **ttRepAgentExitingTrap**
- **ttRepAgentDiedTrap**
- **ttRepAgentStartingTrap**

These traps are described in "Diagnostics through SNMP Traps" in *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

4

Troubleshooting Replication

This chapter describes how to troubleshoot some of the problems you may encounter when replicating data stores.

This chapter includes the following topics:

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
Access Control is enabled and you do not have DDL privileges	If Access Control is enabled on the data store, you must have DDL privileges to use the CREATE REPLICATION or DROP REPLICATION statements.
Incorrect data store name, host name, or element name.	Check CREATE REPLICATION statement for typos.
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 data stores. Use CREATE TABLE to create tables on the data store, or use the **ttRepAdmin** `-duplicate` utility or the **ttRepDuplicateEx** C function to duplicate the entire data store to be replicated.
Other problems	Review the procedures and requirements described in "Defining Replication Schemes".

Unable to Alter 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
Access Control is enabled and you do not have DDL privileges	If Access Control is enabled on the data store, you must have DDL privileges to use the ALTER REPLICATION statement.
Replication agent 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 data stores, alter the replication scheme on both master and subscriber data stores, then restart both replication agents.
Replication scheme replicates a DATASTORE element	ALTER REPLICATION can only be used for replication schemes that specify TABLE elements. You cannot use ALTER REPLICATION when replicating DATASTORE elements.
Incorrect data store name, host name, or element name	Check ALTER REPLICATION statement for typos.
Replication table defined in the ALTER REPLICATION statement do not exist	Use CREATE TABLE to create table on the data store.
Other problems	Review the procedures and requirements described in "Altering Replication".

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
Access Control is enabled and you do not have ADMIN privileges	If Access Control is enabled on the data store, you must have root or ADMIN privileges to use the **ttAdmin** utility or the **ttRepStart**() or **ttRepStop**() procedures to start or stop a replication agent.
TimesTen daemon not started	Check the state of the TimesTen daemon, as described in "Check the current state of TimesTen processes". If necessary, start the TimesTen daemon as described in "Working with the Oracle TimesTen Data Manager Daemon".
Data store does not participate in a replication scheme.	If a data store does not participate in a replication scheme, attempts to start a replication agent for that data store will fail. Use CREATE REPLICATION to create a replication scheme for the data store.

Using SNMP Traps for Notification of Replication Events

TimesTen can send SNMP traps for certain replication events to enable network management software to take immediate action. The possible SNMP traps that can be sent by TimesTen are:

- **ttRepAgentExitingTrap**
- **ttRepAgentDiedTrap**
- **ttRepAgentStartingTrap**
- **ttRepCatchupStartTrap**
- **ttRepCatchupStopTrap**
- **ttRepReturnTransitionTrap**
- **ttRepSubscriberFailedTrap**
- **ttRepUpdateFailedTrap**

These traps are described in "Diagnostics through SNMP Traps"" in the *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

Replication does not work

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

Possible cause	What to do
TimesTen daemon and/or replication agents not running	Check status of TimesTen daemon and replication agents
Master and subscriber agents not communicating	Check that replication agents are communicating
Replication not in **Start** state	Check replication state
Error in replication scheme	Check replication scheme configuration
Inconsistent owner names for replication scheme and tables	Check owner names
Inconsistent replication tables	Check consistency between replicated tables

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 data stores. The output from a simple replication scheme using a single master and subscriber data store 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 *TimesTen to TimesTen 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, as described in TimesTen Installation" in the *Oracle TimesTen In-Memory Database Installation Guide*.

Example 4.1

```
> ttStatus
TimesTen status report as of Tue Oct 28 10:36:11 2003

Daemon pid 3396 port 15000 instance MYINSTANCE
TimesTen server pid 3436 started on port 15002
```

```
------------------------------------------------------------------------
Data store c:\temp\subscriberds
There are 3 connections to the data store
Data store is in shared mode
Shared Memory KEY Global\DBI3f9eb597.1.SHM.6 HANDLE 0x304
Subdaemon pid 1336 context 0x5157d8 connected (KEY Global\DBI3f9eb597.1.SHM.6)
Replication pid 2988 context 0xbc4068 connected (KEY Global\DBI3f9eb597.1.SHM.6)
Replication pid 2988 context 0xbe7720 connected (KEY Global\DBI3f9eb597.1.SHM.6)
cache agent restart policy: manual
------------------------------------------------------------------------
Data store c:\temp\masterds
There are 4 connections to the data store
Data store is in shared mode
Shared Memory KEY Global\DBI3f9eb58b.0.SHM.8 HANDLE 0x2e0
Subdaemon pid 2364 context 0x5157d8 connected (KEY Global\DBI3f9eb58b.0.SHM.8)
Replication pid 2496 context 0xbc4018 connected (KEY Global\DBI3f9eb58b.0.SHM.8)
Replication pid 2496 context 0xbe76d0 connected (KEY Global\DBI3f9eb58b.0.SHM.8)
Replication pid 2496 context 0xb97a90 connected (KEY Global\DBI3f9eb58b.0.SHM.8)
cache agent restart policy: manual
```

Example 4.2

```
> ttStatus
TimesTen status report as of Tue Oct 28 10:31:30 2003

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

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

Note: On UNIX systems, if the TimesTen daemon is not running, **ttStatus** reports “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* data store is replicating to *subscriberds*, the output should look like:

Example 4.4

```
> ttRepAdmin -receiver -list -connStr "dsn=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
```

Next, use the **ttDaemonLog** utility to check the daemon log for errors:

```
> ttDaemonLog | grep Err > Errors
```

Check replication state

Use the **ttRepDeactivate()** procedure to check state of the subscriber data store with respect to its master. If the subscriber is in the **Stop**, **Pause**, or **Failed** state, you can use the **ttRepDeactivate()** procedure to reset the subscriber state to **Start**, as described in "Setting the replication state of subscribers".

Example 4.5

Use **ttRepDeactivate()** to obtain the status of the *subscriberds* data store from its master data store, *masterDSN*, enter:

```
> ttIsql -connStr "dsn=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, as described in "Setting the replication state of subscribers" in the *TimesTen to TimesTen Replication Guide*.
- Do the reported **Peer names** match the names given in the **DataStore** attributes in the DSN definitions for the replicated data stores? Replication will 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 data store 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".
- Are the correct table names displayed under **Table details**? If not, correct the table names in your replication scheme configuration file.

Example 4.6

```
> ttRepAdmin -showconfig -connStr "dsn=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

You can check the TTREP.TTSTORES table to confirm that replication associates the replication scheme with the local data store. Connect to the data store and enter:

```
select * from ttrep.ttstores where is_local_store <> 0x0;
```

Example 4.7

```
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 support. If no rows are returned, then none of the hosts returned by:

```
select distinct(host_name) from ttrep.ttstores;
```

is perceived to be a local system by TimesTen replication. It may also be that none of the data store 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 replication scheme are wrong or misspelled
- Host names or IP addresses cannot be resolved or found by dns or in /etc/hosts
- Entries in the `/etc/hosts` file are incorrectly ordered in appearance. This error is most common when multiple NICs are used. It is strongly recommended to have a Network or System Administrator make changes to `/etc/hosts` files or DNS configuration.

See "Configuring host IP addresses" in the *TimesTen to TimesTen Replication Guide* for details on how to configure DNS and `/etc/hosts` files for host machines used for replication.

To check if a host name in the replication scheme matches the hostname of the local machine, follow these steps:

1. Call **gethostname()** to determine the host name of the running host.
2. Call **gethostbyname()** with the output of from Step 1.
3. Call **gethostbyname()** with the host name specified in the replication scheme.
4. Union output of Step 1 and Step 2. 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" and "Owner of the replication scheme and tables" in the *TimesTen to TimesTen Replication Guide*, the owner names of your replication scheme and your replicated tables must be consistent across all participating data stores.

Checking replication owner

You can 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 data stores 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 On SYSTEM1:

```
> ttIsql -connStr "dsn=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 On SYSTEM1:

```
> ttIsql -connStr "dsn=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

You can check the owner names assigned to the tables in each data store by using the **ttIsql** command, **tables**.

Example 4.10 shows that the TAB table has a consistent owner name (REPL) in the data stores on both SYSTEM1 and SYSTEM2. Example 4.11 shows the TAB table with inconsistent owner names, which were automatically assigned for each host.

Example 4.10

SYSTEM1

```
Command> tables;
  SYS.CACHE_GROUP
  SYS.COLUMNS
  SYS.COL_STATS
  SYS.INDEXES
  SYS.MONITOR
  SYS.PLAN
  SYS.TABLES
  SYS.TBL_STATS
  SYS.TRANSACTION_LOG_API
  REPL.TAB
  TTREP.REPELEMENTS
  TTREP.REPLICATIONS
  TTREP.REPPEERS
  TTREP.REPSTORES
  TTREP.REPSUBSCRIPTIONS
  TTREP.REPTABLES
  TTREP.TTSTORES
17 tables found.
```

SYSTEM2

```
Command> tables;
  SYS.CACHE_GROUP
  SYS.COLUMNS
  SYS.COL_STATS
  SYS.INDEXES
  SYS.MONITOR
  SYS.PLAN
  SYS.TABLES
  SYS.TBL_STATS
  SYS.TRANSACTION_LOG_API
  REPL.TAB
  TTREP.REPELEMENTS
  TTREP.REPLICATIONS
  TTREP.REPPEERS
  TTREP.REPSTORES
  TTREP.REPSUBSCRIPTIONS
  TTREP.REPTABLES
  TTREP.TTSTORES
17 tables found.
```

Example 4.11

SYSTEM1

```
Command> tables;
  SYS.CACHE_GROUP
  SYS.COLUMNS
  SYS.COL_STATS
  SYS.INDEXES
  SYS.MONITOR
  SYS.PLAN
  SYS.TABLES
  SYS.TBL_STATS
  SYS.TRANSACTION_LOG_API
  SYSTEM1.TAB
  TTREP.REPELEMENTS
  TTREP.REPLICATIONS
  TTREP.REPPEERS
  TTREP.REPSTORES
  TTREP.REPSUBSCRIPTIONS
  TTREP.REPTABLES
  TTREP.TTSTORES
17 tables found.
```

SYSTEM2

```
Command> tables;
  SYS.CACHE_GROUP
  SYS.COLUMNS
  SYS.COL_STATS
  SYS.INDEXES
  SYS.MONITOR
  SYS.PLAN
  SYS.TABLES
  SYS.TBL_STATS
  SYS.TRANSACTION_LOG_API
  SYSTEM2.TAB
  TTREP.REPELEMENTS
  TTREP.REPLICATIONS
  TTREP.REPPEERS
  TTREP.REPSTORES
  TTREP.REPSUBSCRIPTIONS
  TTREP.REPTABLES
  TTREP.TTSTORES
17 tables found.
```

Check consistency between replicated tables

Replicated on both master and subscriber data stores must be exactly the same.

Replication Unresponsive, Appears Hung

Possible cause	What to do
Failed subscriber	See "Check replication state".
Return-receipt timeout period too long	See "Check return-receipt timeout setting".

Check replication state

Use the **ttRepDeactivate()** procedure to check state of the subscriber data store with respect to its master. If the subscriber is in the **Failed** state, see "Managing data store failover and recovery" for information on how to deal with failed data stores.

Example 4.12 Use **ttRepDeactivate()** to obtain the status of the *subscriberds* data store from its master data store, *masterDSN*, enter:

```
> ttIsql -connStr "dsn=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 acknowledgement 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 "Checking the status of return service transactions" 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 log files on disk, can impact the performance of both replication and XLA applications.

Possible cause	What to do
Slow network	See "Check network bandwidth"
Using RETURN RECEIPT	See "Check use of return-receipt blocking"
Inefficient replication scheme	See "Check replication configuration"
Log buffer too small	See "Check size of log buffer"
Frequent or inefficient disk writes	See "Check durability settings"
Reading from log files on disk rather than the log buffer	See "Check for reads from log files"

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" 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, but not all, 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” under "Using a return service" for details.

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

Check replication configuration

In addition to return-receipt setting described above, a number of other factors related to the configuration of your replication scheme could impact replication performance. As described in "Performance and recovery trade-offs", you often have to weigh the ability to efficiently failover and recover a data store against replication performance.

Topics that might be of interest include:

- "Efficiency and economy" in *TimesTen to TimesTen Replication Guide*
- "Direct replication or propagation" in *TimesTen to TimesTen Replication Guide*

Check size of log buffer

As described in "Setting attributes for disk-based logging", setting your log buffer too small may impact replication performance. Try setting the **LogBuffSize** DSN attribute to a larger size.

Check durability settings

you can improve replication performance by setting TRANSMIT NONDURABLE on the replication ELEMENT to eliminate the flush-log-to-disk operation from the replication cycle described in "How replication works". See "Setting transmit durability on data store elements" for details.

Enabling the DURABLE COMMIT option in your replication scheme also impacts performance. See "DURABLE COMMIT" for more information.

Check for reads from 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 data store. Normally, replication and XLA readers get update records from the log buffer in memory. When the readers fall behind the application update rate, log files can accumulate on the disk until the backlog can be cleared. This forces the readers to read transactions from the log files on disk, which is much slower. Should you detect reads from the 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 log files on disk 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 data store, 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 log files.

For more complete monitoring of replication progress, you can 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 4.13 For example, you name the above script *monitorLog* and your replication scheme replicates from the MASTERDSN data store to the SUBSCRIBER1DSN data store. You can then check the status of the transaction log by entering:

```
$ monitorLog masterdsn
```

This will generate output similar to:

```
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 will output an updated status every 15 seconds until you Control-C to exit.

Following the date in the output in Example 4.13 is the name of the subscriber, its host, and so on. Next is latency and rate information, as well as the number of log files being retained on behalf of this subscriber. The specific meaning of each

value is described in "From the command line: ttRepAdmin -receiver -list". 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 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 4.13. A steadily increasing the ‘Logs’ value indicates latency is increasing and eventually log reads will be satisfied from disk.

Note: If the **LogBuffSize** is larger than the **LogFileSize**, an increase in the ‘Logs’ value does not necessarily mean the log readers are reading from the 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 **LogBuffSize** 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 disk.

The output from:

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

displays the number of the log file(s) and the location of the bookmarks set by the log manager, as described in "From the command line: ttRepAdmin -bookmark". 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 4.14 In this example, assume the **LogBuffSize** is 16MB than 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 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

Problems using ttRepAdmin -duplicate

If you connected to your new subscriber DSN before running **ttRepAdmin** `-duplicate`, the data store will have already been created. In this situation, `-duplicate` returns:

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

You can confirm the existence of the data store by running **ttStatus** and checking to see if the data store is in the returned list. If the new subscriber data store exists, destroy it and try **ttRepAdmin** `-duplicate` again:

```
> ttDestroy /tmp/newstore
```

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

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

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

If you see errors, such as “unable to copy/create file,” it indicates a lack of space in the temporary directories. The following are the directories where the temporary files are stored.

Operating System	/tmp file Location
Solaris	/tmp
Windows	c:\temp
Linux	/tmp
HP-UX	/var/tmp
AIX	/tmp

You can override the temporary directories by defining the variable TMP with the location on Windows, and with TMPDIR on all other platforms. This variable needs to define a valid directory in the root environment (system in Windows) of the sending machine and the user environment on the receiving machine.

To make changes to the TMP or TMPDIR variable visible to the TimesTen daemon:

- On Red Hat Linux, update the daemon startup script (/etc/rc.d/init.d/tt_*instance-name*) to redirect the temporary directory.

- On Solaris, update the daemon startup script (/etc/rc2.d/S90tt_*instance-name*) to redirect the temporary directory.
- On AIX, use `startsrc` with the `-e` option to set the environment variable. For example:

```
startsrc -e 'TMPDIR=<temp-dir>' -s tt_<instance-name>
```

Note: Stopping and restarting the TimesTen daemon will invalidate all TimesTen datastores currently in memory, so it should not be done when applications are connected to any of these datastores.

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 data store, use the **ttIsql** utility to connect, and enter:

```
Command> SELECT * from TTREP.REPLICATIONS;
```

Example 4.15 This example shows that two replications schemes, REPSCHEME1 and REPSCHEME2, are assigned to the data store 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 configuring CHECK CONFLICTS

When attempting to set CHECK CONFLICTS for an ELEMENT in a CREATE REPLICATION statement, you may encounter an error that states something like:

```
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.

You may receive an error similar to:

```
2208: Column TS does not exist in table.
```

In this situation, confirm you have specified the correct name for the timestamp COLUMN in your 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.

Index

U