Oracle® TimesTen In-Memory Database

Reference

Release 11.2.1

E13069-07

October 2010

ORACLE®

Oracle TimesTen In-Memory Database Reference, Release 11.2.1

E13069-07

Copyright © 1996, 2010, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface ix

Audience ix
Related documents ix
Conventions ix
Documentation Accessibility x
Technical support xi

What's New xiii

New features in Release 11.2.1.6.0 xiii
New features for Release 11.2.1.5.0 xiii
New features for Release 11.2.1.4.0 xiii
New features for Release 11.2.1.1.0 xiv

1 Connection Attributes

Required privileges for attributes 1-2
List of Attributes 1-2
Data store attributes 1-9
Data Source Name 1-10
DataStore 1-11
DatabaseCharacterSet 1-12
Supported character sets 1-12
See also 1-14
Description 1-15
LogDir 1-16
Preallocate 1-17
ReplicationApplyOrdering 1-18
ReplicationParallelism 1-19
Temporary 1-20
TypeMode 1-21
First connection attributes 1-22
AutoCreate 1-23
CkptFrequency 1-24
CkptLogVolume 1-26
CkptRate 1-27
Connections 1-28

ForceConnect 1-29
LogAutoTruncate 1-30
LogBufMB 1-31
LogBufParallelism 1-32
LogFileSize 1-33
LogFlushMethod 1-34
Logging 1-35
LogPurge 1-36
MemoryLock 1-37
Overwrite 1-39
PermSize 1-40
ReceiverThreads 1-41
RecoveryThreads 1-42
TempSize 1-43
General connection attributes 1-44
ConnectionName 1-45
DDLCommitBehavior 1-46
Diagnostics 1-49
DuplicateBindMode 1-50
DurableCommits 1-51
Isolation 1-52
LockLevel 1-54
LockWait 1-55
MatchLogOpts 1-56
PermWarnThreshold 1-57
PrivateCommands 1-58
PWDCrypt 1-59
QueryThreshold 1-60
ReplicationTrack 1-61
SQLQueryTimeout 1-62
TempWarnThreshold 1-63
UID and PWD 1-64
WaitForConnect 1-65
NLS general connection attributes 1-66
ConnectionCharacterSet 1-67
NLS_LENGTH_SEMANTICS 1-68
NLS_NCHAR_CONV_EXCP 1-69
NLS_SORT 1-70
Supported linguistic sorts 1-71
PL/SQL first connection attributes 1-74
PLSQL 1-75
PLSQL_MEMORY_ADDRESS 1-77
PLSQL_MEMORY_SIZE 1-79
PL/SQL general connection attributes 1-80
PLSCOPE_SETTINGS 1-81
PLSQL_CCFLAGS 1-82
PLSQL_CONN_MEM_LIMIT 1-83

PLSQL_OPTIMIZE_LEVEL 1-84
PLSQL_TIMEOUT 1-85
IMDB Cache first connection attributes 1-86
CacheAWTMethod 1-87
IMDB Cache database attributes 1-88
CacheGridEnable 1-89
CacheGridMsgWait 1-90
IMDB Cache general connection attributes 1-91
DynamicLoadEnable 1-92
DynamicLoadErrorMode 1-93
OracleNetServiceName 1-94
OraclePWD 1-95
PassThrough 1-96
RACCallback 1-100
TimesTen Client connection attributes 1-101
TCP_Port 1-102
TCP_Port2 1-103
TTC_FailoverPortRange 1-104
TTC_Server 1-105
TTC_Server2 1-106
TTC_Server_DSN 1-107
TTC_Server_DSN2 1-108
TTC_Timeout 1-109
Server connection attributes 1-111
MaxConnsPerServer 1-112
ServersPerDSN 1-113
ServerStackSize 1-114

2 Built-In Procedures

ttAgingLRUConfig 2-2
ttAgingScheduleNow 2-5
ttApplicationContext 2-7
ttBackupStatus 2-8
ttBlockInfo 2-10
ttBookmark 2-11
ttCacheAutorefresh 2-12
ttCacheAutorefreshStatsGet 2-14
ttCacheAwtMonitorConfig 2-18
ttCacheAWTThresholdGet 2-20
ttCacheAWTThresholdSet 2-21
ttCacheConfig 2-22
ttCacheDbCgStatus 2-26
ttCacheDDLTrackingConfig 2-28
ttCachePolicyGet 2-29
ttCachePolicySet 2-31
ttCachePropagateFlagSet 2-33
ttCacheSqlGet 2-34

ttCacheStart 2-36
ttCacheStop 2-37
ttCacheUidGet 2-38
ttCacheUidPwdSet 2-39
ttCkpt 2-40
ttCkptBlocking 2-42
ttCkptConfig 2-44
ttCkptHistory 2-47
ttCompact 2-51
ttCompactTS 2-52
ttConfiguration 2-53
ttContext 2-54
ttDataStoreStatus 2-55
ttDurableCommit 2-57
ttGridAttach 2-58
ttGridCheckOwner 2-60
ttGridCreate 2-61
ttGridDestroy 2-62
ttGridDetach 2-63
ttGridDetachAll 2-64
ttGridDetachList 2-65
ttGridGlobalCGResume 2-66
ttGridGlobalCGSuspend 2-67
ttGridInfo 2-68
ttGridNameSet 2-70
ttGridNodeStatus 2-71
ttHostNameGet 2-73
ttHostNameSet 2-74
ttLockLevel 2-75
ttLockWait 2-76
ttLogHolds 2-78
ttMonitorHighWaterReset 2-80
ttOptClearStats 2-81
ttOptEstimateStats 2-83
ttOptGetColStats 2-85
ttOptGetFlag 2-86
ttOptGetMaxCmdFreeListCnt 2-87
ttOptGetOrder 2-88
ttOptSetColIntvlStats 2-89
ttOptSetColStats 2-92
ttOptSetFlag 2-94
ttOptSetMaxCmdFreeListCnt 2-98
ttOptSetMaxPriCmdFreeListCnt 2-99
ttOptSetOrder 2-100
ttOptSetTblStats 2-102
ttOptShowJoinOrder 2-104
ttOptUpdateStats 2-106

ttOptUseIndex 2-109
ttPLSQLMemoryStats 2-111
ttRamPolicyGet 2-113
ttRamPolicySet 2-115
ttRedundantIndexCheck 2-117
ttRepDeactivate 2-119
ttReplicationStatus 2-120
ttRepPolicyGet 2-122
ttRepPolicySet 2-124
ttRepQueryThresholdGet 2-126
ttRepQueryThresholdSet 2-127
ttRepStart 2-129
ttRepStateGet 2-130
ttRepStateSave 2-133
ttRepStateSet 2-135
ttRepStop 2-137
ttRepSubscriberStateSet 2-138
ttRepSubscriberSync 2-140
ttRepSubscriberWait 2-142
ttRepSyncGet 2-144
ttRepSyncSet 2-147
ttRepSyncSubscriberStatus 2-150
ttRepTransmitGet 2-151
ttRepTransmitSet 2-152
ttRepXactStatus 2-154
ttRepXactTokenGet 2-156
ttSetUserColumnID 2-158
ttSetUserTableID 2-159
ttSize 2-160
ttSQLCmdCacheInfo 2-162
ttSQLCmdCacheInfoGet 2-164
ttSQLCmdQueryPlan 2-165
ttVersion 2-168
ttWarnOnLowMemory 2-169
ttXactIdGet 2-170
ttXlaBookmarkCreate 2-171
ttXlaBookmarkDelete 2-172
ttXlaSubscribe 2-173
ttXlaUnsubscribe 2-174

3 Utilities

Overview 3-1
Required authentication and authorization for utilities 3-1
Required user authentication for utilities 3-1
Required privileges for executing utilities 3-2
ttAdmin 3-3
ttAdoptStores 3-9

ttBackup 3-11
ttBulkCp 3-14
ttCapture 3-27
ttCheck 3-29
ttCWAdmin 3-32
ttDaemonAdmin 3-36
ttDaemonLog 3-38
ttDaemonLog for Windows 3-39
ttDaemonLog for UNIX 3-42
ttDestroy 3-45
ttIsql 3-47
ttMigrate 3-71
ttmodinstall 3-86
ttRepAdmin 3-88
Help and version information 3-90
Database information 3-91
Subscriber database operations 3-92
Duplicate a database 3-94
Wait for updates to complete 3-98
Replication status 3-99
ttRestore 3-102
ttSchema 3-104
ttSize 3-108
ttStatus 3-110
ttSyslogCheck (UNIX) 3-113
ttTail 3-114
ttTraceMon 3-115
ttuser 3-118
ttVersion 3-119
ttXactAdmin 3-121
ttXactLog 3-128

Index

Preface

Oracle TimesTen In-Memory Database is a memory-optimized relational database. Deployed in the application tier, Oracle TimesTen In-Memory Database operates on databases that fit entirely in physical memory using standard SQL interfaces. High availability for the in-memory database is provided through real-time transactional replication.

Audience

This document provides a reference for TimesTen attributes, built-in procedures, and utilities.

This document is intended for readers with a basic understanding of database systems.

Related documents

TimesTen documentation is available on the product distribution media and on the Oracle Technology Network:

http://www.oracle.com/technetwork/database/timesten/documentation/

Conventions

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, and AIX.

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

This document uses the following text conventions:

Convention	Meaning
italic	Italic type indicates terms defined in text, book titles, or emphasis.
`monospace`	Monospace type indicates code, commands, URLs, function names, attribute names, directory names, file names, text that appears on the screen, or text that you enter.

Convention	Meaning
italic monospace	Italic monospace type indicates a placeholder or a variable in a code example for which you specify or use a particular value. For example: `Driver=install_dir/lib/libtten.sl` Replace *install_dir* with the path of your TimesTen installation directory.
[ ]	Square brackets indicate that an item in a command line is optional.
{ }	Curly braces indicated that you must choose one of the items separated by a vertical bar (|) in a command line.
|	A vertical bar (or pipe) separates alternative arguments.
. . .	An ellipsis (. . .) after an argument indicates that you may use more than one argument on a single command line. An ellipsis in a code example indicates that what is shown is only a partial example.
%	The percent sign indicates the UNIX shell prompt.

In addition, TimesTen documentation uses the following special conventions:

Convention	Meaning
install_dir	The path that represents the directory where TimesTen is installed.
TTinstance	The instance name for your specific installation of TimesTen. Each installation of TimesTen must be identified at installation time with a unique instance name. This name appears in the installation path.
bits or *bb*	Two digits, either 32 or 64, that represent either a 32-bit or 64-bit operating system.
release or *rr*	Numbers that represent a major TimesTen release, with or without dots. For example, 1121 or 11.2.1 represents TimesTen Release 11.2.1.
DSN	TimesTen data source name (for the TimesTen database).

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Technical support

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

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

What's New

This section summarizes the new features of Oracle TimesTen In-Memory Database release 11.2.1 that are documented in this guide and provides links to more information.

New features in Release 11.2.1.6.0

- The system limits are now documented in *Oracle TimesTen In-Memory Database System Tables and Limits Reference*.
- You can perform a global query on cache tables and noncache tables across all nodes in a cache grid. Also, you can unload a cache group on all grid members by specifying a global unload operation. See the description of the optimizer hint "`GlobalProcessing`" on page 2-95.
- If your application has predictable transactional dependencies and does not require the commit order on the target database be the same as the order on the source database, then you can increase replication throughput by configuring parallel replication. See the descriptions for these new attributes:
  - "ReplicationApplyOrdering" on page 1-18.
  - "ReplicationParallelism" on page 1-19.
  - "ReplicationTrack" on page 1-61.

New features for Release 11.2.1.5.0

- Additional support for Clusterware. New options are available with the ttCWAdmin utility that allow users to add and delete master and subscriber hosts dynamically.

  See the ttCWAdmin utility.

New features for Release 11.2.1.4.0

- PL/SQL

  See "PL/SQL first connection attributes" and "PL/SQL general connection attributes"in Chapter 1, "Connection Attributes."
- Additional cache features

  See the ttCacheAutorefresh built-in procedure.
- Additional Cache grid features.

See the ttGridDetachAll, ttGridGlobalCGResume, and ttGridGlobalCGSuspend built-in procedures.

- Support for synonyms

  See ttIsql utility and the PassThrough connection attribute.

New features for Release 11.2.1.1.0

- Cache grid.

  See the CacheGridEnable and CacheGridMsgWait connection attributes and the ttGridAttach, ttGridCreate, ttGridDestroy, ttGridDetach, ttGridDetachList, ttGridInfo, ttGridNameSet and ttGridNodeStatus built-in procedures.

- Dynamic cache groups.

  See the DynamicLoadEnable and DynamicLoadErrorMode connection attributes and the ttCacheSqlGet built-in procedure.

- Client/Server automatic failover.

  See "TimesTen Client connection attributes" in Chapter 1, "Connection Attributes."

- Additional passthrough features.

  See the PassThrough connection attribute, the ttOptSetFlag built-in procedure and the ttIsql utility.

- Oracle-style DDL commit behavior.

  See the DDLCommitBehavior connection attribute.

- SQL command cache.

  See the ttSQLCmdCacheInfoGet and ttSQLCmdQueryPlan built-in procedures.

- Parallel log manager.

  See the LogBufParallelism first connection attribute.

New features specific to support for PL/SQL in TimesTen:

- New connection attributes to support and configure PL/SQL in TimesTen.

  See PL/SQL first connection attributes and PL/SQL general connection attributes.

- New ttIsql commands to display PL/SQL functions, packages, procedures and more.

  See the ttIsql utility.

New features specific to replication:

- Support for Oracle Clusterware.

  See the ttCWAdmin utility.

- Replication query threshold settings.

  See the ttRepQueryThresholdGet and ttRepQueryThresholdSet built-in procedures and the ttAdmin utility.

1

Connection Attributes

The ODBC standard defines four connection attributes:

- DSN
- Driver
- UID
- PWD

WINDOWS

For a description of the ODBC definition of these attributes, see the appropriate ODBC manual for your platform:

- *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide*
- *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide*

This chapter describes all the attributes defined by TimesTen. To view the names and values of most attributes specified in the connection string, an application can use the `ttConfiguration` built-in procedure.

> **Note:** According to the ODBC standard, when an attribute occurs multiple times in a connection string, the first value specified is used, not the last value.

UNIX

On UNIX, `False` means the attribute value is set to `0` and `True` means the attribute value is set to `1`.

WINDOWS

On Windows, `False` means the check box is unchecked and `True` means the check box is checked.

The following sections provide details on all TimesTen attributes, which are first listed in tables in "List of Attributes" on page 1-2. Following the tables, each attribute is described in detail.

- Required privileges for attributes
- List of Attributes
- Data store attributes
- First connection attributes
- General connection attributes
- NLS general connection attributes
- PL/SQL first connection attributes

- PL/SQL general connection attributes
- IMDB Cache first connection attributes
- IMDB Cache database attributes
- IMDB Cache general connection attributes
- Client connection attributes
- Server connection attributes

Required privileges for attributes

Only the instance administrator can change a first connection attribute to a value other than the one currently in effect. (No privileges are required to change `AutoCreate` and `ForceConnect`.)

List of Attributes

This section includes the tables:

- Table 1–1, " Data store attributes"
- Table 1–2, " First connection attributes"
- Table 1–3, " General connection attributes"
- Table 1–4, " NLS general connection attributes"
- Table 1–5, " PL/SQL first connection attributes"
- Table 1–6, " PL/SQL general connection attributes"
- Table 1–8, " IMDB Cache database attributes"
- Table 1–10, " Client connection attributes"
- Table 1–11, " Server connection attributes"

Table 1–1 Data store attributes

Name	Description	Default
`Data Source Name`	A name that identifies the specific attributes of a connection to the database.	None
`DataStore`	Identifies the physical database.	None
`DatabaseCharacterSet`	Identifies the character set used by the database. This attribute is required at database creation time.	None
`Description`	A statement that identifies the use of the data source name.	None
`LogDir`	The directory where transaction log files are stored.	Database directory
`Preallocate`	Specifies that disk space for the database should be preallocated when creating the database.	0 (false)

Table 1–1 (Cont.) Data store attributes

Name	Description	Default
ReplicationApplyOrdering	Begins parallel replication.	0 (off)
ReplicationParallelism	Specifies the number of tracks available for parallel replication.	1 (single threaded replication)
Temporary	Specifies that the database is not saved to disk.	0 (false)
TypeMode	The type mode for the database.	0 - Oracle Type Mode

Table 1–2 First connection attributes

Name	Description	Default
AutoCreate	Specifies that the first connection creates the database if it does not exist.	1 (true)
CkptFrequency	Controls the frequency in seconds that TimesTen performs a background checkpoint.	600
CkptLogVolume	Controls the amount of data in megabytes that collects in the log between background checkpoints.	0 (off)
CkptRate	Controls the maximum rate at which data should be written to disk during a checkpoint operation.	0 (unlimited rate)
Connections	Indicates the expected upper bound on the number of concurrent connections to the database.	64
ForceConnect	Specifies whether a connection is allowed to a failed database if it is not properly restored from the corresponding subscriber database.	0 (Connection disallowed)
LogAutoTruncate	Determines whether the first connection to a database should proceed if TimesTen recovery encounters a defective log record	1 (Continues after log is truncated)
LogBufMB	The size of the internal log buffer in MB.	64
LogBufParallelism	The number of log buffer strands.	2
LogFileSize	The transaction log file size in MB.	64

Table 1–2 (Cont.) First connection attributes

Name	Description	Default
LogFlushMethod	Controls the method used by TimesTen to write and sync log data to transaction log files	1 (Write data to transaction log files using buffered writes. Use explicit sync operations as needed to sync log data to disk)
Logging	Specifies what type of logging should be performed for the database. Only logging to disk is supported.	1 (Logging to disk)
LogPurge	Specifies that unneeded transaction log files are deleted during a checkpoint operation.	1 (true)
MemoryLock	Allows applications that connect to a shared database to specify whether the real memory should be locked during database loading	0 (Do not acquire a memory lock)
Overwrite	Specifies that the existing database should be overwritten with a new one when a connection is attempted.	0 (false)
PermSize	The size in MB for the permanent partition of the database.	32
ReceiverThreads	Controls the number of threads used to apply changes on the active master database to the standby master database in an active standby pair replication scheme.	1
RecoveryThreads	The number of threads used to rebuild indexes during recovery.	1
TempSize	The size in MB for the temporary partition of the database.	The default size is determined from the PermSize value.

Table 1–3 General connection attributes

Name	Description	Default
ConnectionName	Specifies whether there is a symbolic name for the data source.	The process name
DDLCommitBehavior	Controls transactional commit behavior in relation to DDL	0 (Oracle behavior)
Diagnostics	Specifies whether diagnostic messages are generated.	1 (Messages are generated.)

Table 1–3 (Cont.) General connection attributes

Name	Description	Default
DuplicateBindMode	Determines whether applications use TimesTen or Oracle parameter binding for duplicate occurrences of a parameter in a SQL statement.	0 (Oracle-style binding)
DurableCommits	Specifies that commit operations should write log records to disk.	0 (Records not written to disk)
Isolation	Specifies whether the isolation level is read committed or serializable.	1 (Read committed)
LockLevel	Specifies whether the connection should use row-level locking (value = 0) or database-level locking (value = 1).	Row-level locking
LockWait	Allows an application to configure the lock wait interval for the connection.	10 seconds
MatchLogOpts	Specifies that values used for the Logging and LogPurge attributes should match those of current connections.	0 (false)
PermWarnThreshold	The threshold at which TimesTen returns a warning and throws an SNMP trap when the permanent partition of the database is low in memory.	90%
PrivateCommands	Determines if commands are shared between connections.	0 (on)
PWD See "UID and PWD" on page 1-64.	Specify the password that corresponds with the specified UID. When caching Oracle data, PWD specifies the TimesTen password. You can specify the Oracle PWD in the connection string, if necessary.	None
PWDCrypt	The value of the encrypted user password.	None
QueryThreshold	Determines whether TimesTen returns an error message and throws an SNMP trap if a query times out before executing.	0 - No error is returned
ReplicationTrack	Assigns a connection to a replication track.	None
SQLQueryTimeout	Specifies the time limit in seconds within which the database should execute SQL statements.	0 (no timeout)

Table 1–3 (Cont.) General connection attributes

Name	Description	Default
TempWarnThreshold	The threshold at which TimesTen returns a warning and throws an SNMP trap when the temporary partition of the database is low in memory.	90%
UID See "UID and PWD" on page 1-64.	Specify a user name that is defined on the server. When caching Oracle data, the UID must match the UID on the Oracle database that is being cached in TimesTen.	None
WaitForConnect	Specifies that the connection attempt should wait if an immediate connection is not possible.	1

Table 1–4 NLS general connection attributes

Name	Description	Default
ConnectionCharacterSet	The character encoding for the connection, which may be different from the database character set.	US7ASCII unless the database character set is TIMESTEN8, then TIMESTEN8.
NLS_LENGTH_SEMANTICS	The default length semantics configuration.	BYTE
NLS_NCHAR_CONV_EXCP	Determines whether an error is reported when there is data loss during an implicit or explicit character type conversion between NCHAR/NVARCHAR data and CHAR/VARCHAR data.	0 (false)
NLS_SORT	The collating sequence to use for linguistic comparisons.	BINARY

Table 1–5 PL/SQL first connection attributes

Name	Description	Default
PLSQL	Determines whether PL/SQL is enabled.	1 (PL/SQL is enabled)
PLSQL_MEMORY_ADDRESS	The virtual address at which the shared memory segment is loaded into each process that uses the TimesTen "direct" drivers.	Platform specific
PLSQL_MEMORY_SIZE	The size in megabytes of the shared memory segment used by PL/SQL.	32 MB

Table 1–6 PL/SQL general connection attributes

Name	Description	Default
PLSCOPE_SETTINGS	Controls whether the PL/SQL compiler generates cross-reference information.	IDENTIFIERS: NONE
PLSQL_CCFLAGS	Controls conditional compilation of PL/SQL units.	NULL
PLSQL_CONN_MEM_LIMIT	Specifies the maximum amount of process heap memory in MB that PL/SQL can use for this connection.	100
PLSQL_OPTIMIZE_LEVEL	The optimization level that is used to compile PL/SQL library units.	2
PLSQL_TIMEOUT	The number of seconds a PL/SQL procedure can run before being automatically terminated.	30 seconds

Table 1–7 IMDB Cache first connection attributes

Name	Description	Default
CacheAWTMethod	Enables the AWT propagation method to be used on Oracle tables.	0 (sql)

Table 1–8 IMDB Cache database attributes

Name	Description	Default
CacheGridEnable	Enables cache grid.	On
CacheGridMsgWait	Sets the maximum message wait time.	60 seconds

Table 1–9 IMDB Cache general connection attributes

Name	Description	Default
DynamicLoadEnable	Enables or disables transparent load of Oracle data to dynamic cache groups.	1 (Dynamic cache group load is enabled)
DynamicLoadErrorMode	Determines if an error message is returned upon a transparent load failure.	0 (Errors are not returned)
OracleNetServiceName	The Oracle Service Name of the Oracle instance from which data is to be loaded into a TimesTen database. This attribute is only used by the cache agent. Set the OracleNetServiceName attribute to the Oracle Service Name.	None
OraclePWD	Identifies the password for the Oracle database that is being cached in TimesTen.	None
PassThrough	Specifies which SQL statements are executed locally in TimesTen and which SQL statements are passed through to Oracle for execution.	0

Table 1–9 (Cont.) IMDB Cache general connection attributes

Name	Description	Default
RACCallback	Specifies whether to enable or disable the installation of Application Failover (TAF) and Fast Application Notification (FAN) callbacks.	1 (Install the callbacks.)

Table 1–10 Client connection attributes

Name	Description	Default
TCP_Port	The port number on which the server is listening.	None
TCP_Port2	The port number on which the server should listen if an automatic failover occurs.	None
TTC_FailoverPortRange	A range for the failover port numbers	None
TTC_Server	Name of the computer where the TimesTen Server is running or a logical server name.	None
TTC_Server2	If an automatic failover occurs, the name of the computer where the TimesTen Server should be running or a logical server name.	None
TTC_Server_DSN	Server DSN corresponding to the TimesTen database.	None
TTC_Server_DSN2	Server DSN corresponding to the TimesTen database, if an automatic failover occurs.	None
TTC_Timeout	Optional. Timeout period, in seconds, for completion of a TimesTen client/server operation. The maximum timeout period is 99999 seconds.	60 seconds

Table 1–11 Server connection attributes

Name	Description	Default
MaxConnsPerServer	The maximum number of concurrent connections a child server process can handle.	1
ServersPerDSN	The desired number of server processes for the DSN.	1
ServerStackSize	The size in KB of the thread stack for each connection.	
	For 32-bit systems:	128 KB
	For 64-bit systems:	256KB

Data store attributes

Data store attributes are set at data store creation time.The data store attributes are listed inTable 1–1, " Data store attributes" and described in detail in this section.

These attributes can be assigned values only during database creation by the instance administrator.

Data Source Name

The data source name uniquely identifies the attributes to a connection. It serves two purposes:

- As a unique identifier to the ODBC driver manager (if one is present), allowing it to associate a Data Store Name with a specific ODBC driver.
- As one of potentially many name aliases to a single physical database where the name alias has unique attributes associated with it.

The database attributes can apply to either the data source name (connection to a database) or the Data Store Path Name (database).

On Windows, the data source name and all configuration information associated with the data source (including the database path name) are stored in the system registry. This information is used by the ODBC driver manager and by TimesTen.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set Data Source Name as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`DSN`	A name that describes the DSN.
Windows ODBC Data Source Administrator	**Data Source Name** field	A name that describes the DSN.

DataStore

The database path name uniquely identifies the physical database. It is the full path name of the database and the file name prefix, for example: `C:\data\AdminData`. This name is not a file name. The actual database file names have suffixes, such as `.ds0` and `.log0`, for example `C:\data\AdminData.ds0` and `C:\data\AdminData.log0`.

You can use environment variables in the specification of the database path and name.

> **Note:** You are required to specify the database path and name at database creation time. It cannot be altered after the database has been created.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set DataStore as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`DataStore`	Full path to the physical database that the data source name references.
Windows ODBC Data Source Administrator	**Data Store Path + Name** field	Full path to the physical database that the data source name references.

DatabaseCharacterSet

The database character set determines the character set in which data is stored.

> **Note:** You are required to specify the database character set at database creation time only. It cannot be altered after the database has been created. If you do not specify a value for this attribute when creating a database, TimesTen returns error message 12701.

Generally, your database character set should be chosen based on the data requirements. For example: Do you have data in Unicode or is your data in Japanese on UNIX (EUC) or Windows (SJIS)?

You should choose a connection character set that matches your terminal settings or data source. See "ConnectionCharacterSet" on page 1-67.

When the database and connection character sets differ, TimesTen performs the data conversion internally based on the connection character set. If the connection and database character sets are the same, TimesTen does not need to convert or interpret the data set. Best performance occurs when connection and database character sets match, since no conversion is required.

To use this attribute you must specify a supported character set. For a list of character set names that can be used as a value for this attribute, see "Supported Character Sets."

There are several things to consider when choosing a character set for your database. For a discussion about these considerations, see "Choosing a database character set" in *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set DatabaseCharacterSet name as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`DatabaseCharacterSet`	Specify the preferred character set.
Windows ODBC Data Source Administrator	**Database Character Set** list	Select the preferred character set from the list provided in the ODBC Data Source Administrator.

Supported character sets

The tables in this section describe the character sets supported in TimesTen.

Asian character sets

Name	Description
JA16EUC	EUC 24-bit Japanese

Name	Description
JA16EUCTILDE	The same as JA16EUC except for the way that the wave dash and the tilde are mapped to and from Unicode
JA16SJIS	Shift-JIS 16-bit Japanese
JA16SJISTILDE	The same as JA16SJIS except for the way that the wave dash and the tilde are mapped to and from Unicode
KO16KSC5601	KSC5601 16-bit Korean
KO16MSWIN949	Microsoft Windows Code Page 949 Korean
TH8TISASCII	Thai Industrial Standard 620-2533 - ASCII 8-bit
VN8MSWIN1258	Microsoft Windows Code Page 1258 8-bit Vietnamese
ZHS16CGB231280	CGB2312-80 16-bit Simplified Chinese
ZHS16GBK	GBK 16-bit Simplified Chinese
ZHS32GB18030	GB18030-2000
ZHT16BIG5	BIG5 16-bit Traditional Chinese
ZHT16HKSCS	Microsoft Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001. Character set conversion to and from Unicode is based on Unicode 3.0.
ZHT16MSWIN950	Microsoft Windows Code Page 950 Traditional Chinese
ZHT32EUC	EUC 32-bit Traditional Chinese

European character sets

Name	Description
BLT8CP921	Latvian Standard LVS8-92(1) Windows/UNIX 8-bit Baltic
BLT8ISO8859P13	ISO 8859-13 Baltic
BLT8MSWIN1257	Microsoft Windows Code Page 1257 8-bit Baltic
BLT8PC775	IBM-PC Code Page 775 8-bit Baltic
CEL8ISO8859P14	ISO 8859-13 Celtic
CL8ISO8859P5	ISO 8859-5 Latin/Cyrillic
CL8KOI8R	RELCOM Internet Standard 8-bit Latin/Cyrillic
CL8KOI8U	KOI8 Ukrainian Cyrillic
CL8MSWIN1251	Microsoft Windows Code Page 1251 8-bit Latin/Cyrillic
EE8ISO8859P2	ISO 8859-2 East European
EL8ISO8859P7	ISO 8859-7 Latin/Greek
ET8MSWIN923	Microsoft Windows Code Page 923 8-bit Estonian
EE8MSWIN1250	Microsoft Windows Code Page 1250 8-bit East European
EL8MSWIN1253	Microsoft Windows Code Page 1253 8-bit Latin/Greek
EL8PC737	IBM-PC Code Page 737 8-bit Greek/Latin
EE8PC852	IBM-PC Code Page 852 8-bit East European
LT8MSWIN921	Microsoft Windows Code Page 921 8-bit Lithuanian
NE8ISO8859P10	ISO 8859-10 North European

Name	Description
NEE8ISO8859P4	ISO 8859-4 North and North-East European
RU8PC866	IBM-PC Code Page 866 8-bit Latin/Cyrillic
SE8ISO8859P3	ISO 8859-3 South European
US7ASCII	ASCII 7-bit American
US8PC437	IBM-PC Code Page 437 8-bit American
WE8ISO8859P1	ISO 8859-1 West European
WE8ISO8859P15	ISO 8859-15 West European
WE8MSWIN1252	Microsoft Windows Code Page 1252 8-bit West European
WE8PC850	IBM-PC Code Page 850 8-bit West European
WE8PC858	IBM-PC Code Page 858 8-bit West European

Middle Eastern character sets

Name	Description
AR8ADOS720	Arabic MS-DOS 720 Server 8-bit Latin/Arabic
AR8ASMO8X	ASMO Extended 708 8-bit Latin/Arabic
AR8ISO8859P6	ISO 8859-6 Latin/Arabic
AR8MSWIN1256	Microsoft Windows Code Page 1256 8-Bit Latin/Arabic
AZ8ISO8859P9E	ISO 8859-9 Latin Azerbaijani
IW8ISO8859P8	ISO 8859-8 Latin/Hebrew
IW8MSWIN1255	Microsoft Windows Code Page 1255 8-bit Latin/Hebrew
TR8MSWIN1254	Microsoft Windows Code Page 1254 8-bit Turkish
TR8PC857	IBM-PC Code Page 857 8-bit Turkish
WE8ISO8859P9	ISO 8859-9 West European & Turkish

TimesTen character set

Name	Description
TIMESTEN8	TimesTen legacy character semantics

Universal character sets

Name	Description
AL16UTF16	Unicode 4.0 UTF-16 Universal character set. This is the implicit TimesTen national character set.
AL32UTF8	Unicode 4.0 UTF-8 Universal character set
UTF8	Unicode 3.0 UTF-8 Universal character set, CESU-8 compliant

See also

"ConnectionCharacterSet" on page 1-67

Description

Optionally, set this attribute to help you identify the Data Source Name (`DSN`) and its attributes.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set Description as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`Description`	Text description of the Data Source Name. This attribute is optional.
Windows ODBC Data Source Administrator	**Description** field	Text description of the Data Source Name. This attribute is optional.

LogDir

The LogDir attribute specifies the directory where database logs reside. Specifying this attribute allows you to place the transaction log files on a different I/O path from the database checkpoint files. This may improve throughput.

You can use environment variables in the specification of the transaction log file path name. For example, you can specify `$HOME/AdminDS` for the location of the database. See "Using environment variables in database path names" in *Oracle TimesTen In-Memory Database Operations Guide* for more information.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `LogDir` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LogDir`	Specifies the directory where transaction log files reside.
Windows ODBC Data Source Administrator	**Transaction Log Directory** field	Specifies the directory where transaction log files reside.

Preallocate

The `Preallocate` attribute determines whether TimesTen preallocates file system space for the database when the database is created. Setting this attribute ensures that there is sufficient space for the database when the database is saved to the file system.

Using Preallocate=1 in combination with ttRestore or ttRepAdmin `-duplicate` and a value of PermSize that does not match the value of PermSize of the original database may result in two checkpoint files with different sizes. This has not been shown to have negative effects. However, the issue can be avoided completely either by using the same PermSize as the original database or by setting Preallocate=0.

When a duplicate operation is carried out, the duplicated store has behavior consistent with a Preallocate setting of 0, even if it is set to 1 on the original or duplicated database. The behavior is indicated by the size of the checkpoint files, which is the sum of the size of the data and size of the database header.

The checkpoint files are subsequently allowed to grow to the same size as checkpoint files on the master database (PermSize + database header), but the space is not preallocated. The checkpoint files increase in size as data is added.

The reason for this behavior is that PreAllocate is set at database creation time. It is not a first connection attribute. The duplicate operation is not a database creation operation, so the preallocate attribute is not honored.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `Preallocate` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	Preallocate	**0** - Does not preallocate file system space for database when creating the database (default). **1** -Preallocates file system space for the database.
Windows ODBC Data Source Administrator	**Preallocate** check box	**unchecked** - Does not preallocate file system space for database when creating the database (default). **checked** - Preallocates file system space for the database.

Note: reallocating disk space for a large database is very time consuming.

ReplicationApplyOrdering

Starts user-specified parallel replication, when used with the `ReplicationParallelism` attribute. Set the value of this attribute to 1, to initiate parallel replication.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `ReplicationApplyOrdering` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`ReplicationApplyOrdering`	**0** - Specifies single threaded replication (default). **1** - Specifies parallel replication.
Windows ODBC Data Source Administrator	**Replication Apply Ordering** check box	**unchecked** - Specifies single threaded replication (default). **checked** - Specifies parallel replication.

ReplicationParallelism

For user-specified parallel replication, this attribute specifies the number of tracks that are replicated in parallel.

You must also set `ReplicationApplyOrdering` to 1 to start parallel replication.

Use the `ReplicationTrack` attribute to assign a track to a connection.

For applications that have very predictable transactional dependencies and do not require that the commit order on the receiver is the same as that on the originating database, you can specify the number of transaction tracks and apply specific transactions to each track. All tracks are read, transmitted and applied in parallel.

The default value for this attribute is 1. If you set the value to be greater than 1, you cannot create a replication scheme using the `CREATE ACTIVE STANDBY PAIR` statement.

When parallel replication is enabled the Description column of the `ttLogHolds` built-in procedure displays one row per track per subscriber node.

Restrictions and things to consider when specifying parallel replication include:

- Parallel replication is available only with regular replication. It is not supported with active standby pair replication.
- Set the value of this attribute to a value that is equal to or smaller than the value of the `LogBufParallelism` attribute. If you specify more replication tracks than log buffer threads, some of the replication tracks remain empty.
- Tables in databases configured for parallel replication cannot be added or dropped.
- Parallel replication is not available for replication with cache groups.
- Synchronous replication, including `TWO SAFE` and `RETURN RECEIPT` replication, is not supported with parallel replication.

To learn more about parallel replication, see the*Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `ReplicationParallelism` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`ReplicationParallelism`	*n* - A value between 1 and 64, indicating the number of tracks to replicate in parallel. The default is 1, single-threaded replication.
Windows ODBC Data Source Administrator	**Parallel Replication** field	*n* - A value between 1 and 64, indicating the number of tracks to replicate in parallel. The default is 1, single-threaded replication.

Temporary

Set this attribute to create a temporary database. Temporary databases are not saved to the file system. They may, however, be shared and therefore require a data store path name. A temporary database is deleted when the last connection is closed. See "Database persistence" in *Oracle TimesTen In-Memory Database Operations Guide* for more information. You cannot assign the Temporary data store attribute to an existing permanent database.

Note: You cannot back up or replicate a temporary database.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `Temporary` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`Temporary`	**0** - Creates permanent database (default). **1** - Creates temporary database.
Windows ODBC Data Source Administrator	**Temporary** check box	**unchecked** - Creates permanent database (default). **checked** - Creates temporary database.

TypeMode

Specifies whether the names and semantics of the data types follow Oracle or TimesTen type rules. TimesTen supports both Oracle and TimesTen data types. The type mode determines what names are used to specify each data type. In some cases, a data type has both an alias name and a fixed type name. In such a situation, you can use either name. The TimesTen type mode is included for backward compatibility. We recommend that you use the default setting, which is Oracle type mode.

When caching Oracle data in TimesTen, `TypeMode` must be set to 0.

See "Type specifications" in *Oracle TimesTen In-Memory Database SQL Reference* for a list of data types and their fixed and alias names.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `TypeMode` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`TypeMode`	0 - Oracle type mode (default) 1 - TimesTen type mode. If no value is specified, either the default type mode or the type mode assigned when the database was created is used.
Windows ODBC Data Source Administrator	**TypeMode** dropdown list	0 - Oracle type mode (default) 1 - TimesTen type mode If no value is specified, either the default type mode or the type mode assigned when the database was created is used.

First connection attributes

First connection attributes are set when a connection is made to an idle database (a database created by the instance administrator which currently has no connections) and persist for that connection and all subsequent connections until the last connection to this database is closed.

First connection attributes are listed in Table 1–2, " First connection attributes" and described in detail in this section.

If you try to connect to the database using attributes that are different from the first connection attribute settings, the new connection may be rejected or the attribute value may be ignored. However, for example, if existing connections have a `LogFileSize` of one size and a new connection specifies a `LogFileSize` of another size, TimesTen ignores the new value and returns a warning.

Note: Only the instance administrator can change a first connection attribute to a value other than the one currently in effect. To change the value of a first connection attribute, you must first shut down the database.

AutoCreate

If you connect to a database that has the `AutoCreate` attribute set and the database does not exist yet, the database is created automatically if you supplied a valid existing path. With `AutoCreate` set, TimesTen creates the database, but not the path to the database. If you attempt to connect to a database that does not exist and the `AutoCreate` attribute is not set, the connection fails.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `AutoCreate` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`AutoCreate`	**0** - Does not create new database if database does not exist. **1** - Creates new database if the specified database does not exist (default).
Windows ODBC Data Source Administrator	**AutoCreate** check box	**unchecked** - Does not create new database if database does not exist. **checked** - Creates new database if database does not exist (default).

CkptFrequency

Controls the frequency in seconds that TimesTen performs a background checkpoint. The counter used for the checkpoint condition is reset at the beginning of each checkpoint.

If both `CkptFrequency` and `CkptLogVolume` attributes have a value greater than 0, a checkpoint is performed when either of the two conditions becomes true. The values set by the `ttCkptConfig` built-in procedure replace the values set by these attributes.

In the case that your application attempts to perform a checkpoint operation while a background checkpoint is in process, TimesTen waits until the background checkpoint finishes and then executes the application's checkpoint. To turn off background checkpointing, set `CkptFrequency`=0 and `CkptLogVolume`=0.

The value of this attribute is "sticky" as it persists across database loads and unloads unless it is explicitly changed. The default value is only used during database creation. Subsequent first connections default to using the existing value stored in the database. If left unspecified (or empty in the Windows ODBC Data Source Administrator), the stored setting is used. To turn the attribute off, you must explicitly specify a value of 0.

Regardless of the value of this attribute, if a checkpoint fails, TimesTen attempts a checkpoint only once every 10 minutes. If a checkpoint failure occurs due to a lack of file system space, we recommend that you attempt a manual checkpoint as soon as space is available. Once any successful checkpoint occurs, background checkpointing reverts to the configured schedule.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `CkptFrequency` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`CkptFrequency`	Enter a value in seconds for the frequency at which TimesTen should perform a background checkpoint. Default is 600 if Logging=1 is specified, otherwise it is 0. To specify the default or "existing" value, leave the value empty. A value of 0 means that checkpoint frequency is not considered when scheduling checkpoints.

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	**Ckpt Frequency (secs)** field	Enter a value in seconds for the frequency at which TimesTen should perform a background checkpoint. Default is 600 if Logging=1 is specified, otherwise it is 0. To specify the default or "existing" value, leave the field empty. A value of 0 means that checkpoint frequency is not considered when scheduling checkpoints.

CkptLogVolume

Controls the amount of data in megabytes that collects in the log between background checkpoints. The counter used for the checkpoint condition is reset at the beginning of each checkpoint.

If both `CkptFrequency` and `CkptLogVolume` attributes have a value greater than 0, a checkpoint is performed when either of the two conditions becomes true. The values set by the `ttCkptConfig` built-in procedure replace the values set by these attributes.

In the case that your application attempts to perform a checkpoint operation while a background checkpoint is in process, TimesTen waits until the background checkpoint finishes and then executes the application's checkpoint. To turn off background checkpointing, set `CkptFrequency`=0 and `CkptLogVolume`=0.

The value of this attribute is "sticky" as it persists across database loads and unloads unless it is explicitly changed. The default value is only used during database creation. Subsequent first connections default to using the existing value stored in the database. If left unspecified (or empty in the Windows ODBC Data Source Administrator), the stored setting is used. To turn the attribute off, you must explicitly specify a value of 0.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `CkptLogVolume` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`CkptLogVolume`	Specify the amount of data in megabytes that can accumulate in the transaction log file between background checkpoints. The default is 0. To specify the default or "existing" value, leave the value empty. A value of 0 means that log volume is not considered when scheduling checkpoints.
Windows ODBC Data Source Administrator	**Ckpt LogVolume** field	Specify the amount of data in megabytes that can accumulate in the transaction log file between background checkpoints. The default is 0. To specify the default or "existing" value, leave the field empty. A value of 0 means that log volume is not considered when scheduling checkpoints.

CkptRate

Controls the maximum rate at which data should be written to disk during a checkpoint operation. This may be useful when the writing of checkpoints to disk interferes with other applications.

This rate is used by all background checkpoints and by checkpoints initiated by the `ttCkpt` and `ttCkptBlocking` built-in procedures. *Foreground checkpoints* (checkpoints taken during first connect and last disconnect) do not use this rate. The rate is specified in MB per second.

A value of 0 disables rate limitation. This is the default. The value can also be specified using the `ttCkptConfig` built-in procedure. The value set by the `ttCkptConfig` built-in procedure replaces the value set by this attribute.

The value of this attribute is "sticky" as it persists across database loads and unloads unless it is explicitly changed. The default value is only used during database creation. Subsequent first connections default to using the existing value stored in the database. If left unspecified (or empty in the Windows ODBC Data Source Administrator), the stored setting is used. To turn the attribute off, you must explicitly specify a value of 0. For existing databases that are migrated to this release, the value is initialized to 0. To use the current or default value, the attribute value should be left unspecified.

For more details about the benefits of and issues when using `CkptRate`, see "Setting the checkpoint rate" in *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `CkptRate` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`CkptRate`	Specify the maximum rate in MB per second at which a checkpoint should be written to disk. A value of 0 indicates that the rate should not be limited. This is the default.
Windows ODBC Data Source Administrator	**CkptRate** field	Specify the maximum rate in MB per second at which a checkpoint should be written to disk. A value of 0 indicates that the rate should not be limited. This is the default.

Connections

Indicates the expected upper bound on the number of concurrent connections to the database. TimesTen allocates one semaphore for each expected connection. If the number of connections exceeds the value of this attribute, the system still operates but may perform suboptimally.

The number of current connections to a database can be determined by viewing the output from the `ttStatus` utility.

A Connections value of 0 or no value indicates that the default number of sempahores should be used. Some TimesTen processes use a database connection. If you receive an error indicating that the number of connections exceeds the value of this attribute, increase the value until you no longer receive this error.

Note: The kernel must be configured with enough semaphores to handle all active databases. For details on setting semaphores for your system, see "Installation prerequisites" in *Oracle TimesTen In-Memory Database Installation Guide*.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `Connections` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`Connections`	no value - Indicates that the default value is used. **0** - Indicates that the default value is used. **64** - Default value **An integer from 1 through 2047** -The value represents the expected maximum number of connections.
Windows ODBC Data Source Administrator	**Connections** field	no value - Indicates that the default value is used. **0** - Indicates that the default value is used. **64** - Default value **An integer from 1 through 2047** -The value represents the expected maximum number of connections.

ForceConnect

When return receipt replication is used with the `NONDURABLE TRANSMIT` option, a failed master database is allowed to recover only by restoring its state from a subscriber database using the `-duplicate` option of the `ttRepAdmin` utility. In other words, the failed database cannot just come up and have replication bring it up to date because it may lose some transactions that were transmitted to the subscriber but not durably committed locally. The `ForceConnect` connection attribute overrides this restriction.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `ForceConnect` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`ForceConnect`	**0** - Do not allow connection to failed database if it is not properly restored from the corresponding subscriber database (default). **1** - Allow connection to a failed database even if it is not properly restored from the corresponding subscriber database.
Windows ODBC Data Source Administrator	**ForceConnect** check box	**unchecked** - Do not allow connection to failed database if it is not properly restored from the corresponding subscriber database (default). **checked** - Allow connection to a failed database even if it is not properly restored from the corresponding subscriber database.

LogAutoTruncate

Determines whether the first connection to the database should proceed if TimesTen recovery encounters a defective log record.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `LogAutoTruncate` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LogAutoTruncate`	**0** - If a defective log record is encountered, terminate recovery and return an error to the connecting application. Checkpoint and transaction log files remain unmodified. **1** - If a defective log record is encountered, truncate the log at the defective record's location and continue with recovery. The original transaction log files are moved to a directory called `savedLogFiles`, which is created as a subdirectory of the log directory. The transaction log files are saved for diagnostic purposes (default).
Windows ODBC Data Source Administrator	**LogAutoTruncate** box	**unchecked** - If a defective log record is encountered, terminate recovery and return an error to the connecting application. Checkpoint and transaction log files remain unmodified. **checked** - If a defective log record is encountered, truncate the log at the defective record's location and continue with recovery. The original transaction log files are moved to a directory called `savedLogFiles`, which is created as a subdirectory of the log directory. The transaction log files are saved for diagnostic purposes (default).

LogBufMB

The `LogBufMB` attribute specifies the size of the internal transaction log buffer in megabytes. The default log buffer size is 64 megabytes.

If you change the value of `LogBufMB`, you also may need to change the value of `LogBufParallelism` to satisfy the constraint that `LogBufMB/LogBufParallelism` >=8.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `LogBufMB` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LogBufMB`	*n* - Size of log buffer in megabytes. If not set and the database exists, the existing value stored in the database is used. If not set and the database is being created, the default value of 64 is used.
Windows ODBC Data Source Administrator	**Log Buffer Size (MB)** field	Size of log buffer, in megabytes. If not set and the database exists, the existing value stored in the database is used. If not set and the database is being created, the default value of 64 is used.

LogBufParallelism

The `LogBufParallelism` attribute specifies the number of transaction log buffer strands to which TimesTen writes log files before the log is written to disk, allowing for improved log performance. Each buffer has its own insertion latch. Records are inserted in any of the strands. The log flusher gathers records from all strands and writes them to the log files.

The maximum number of strands is 64. The default is 4.

If you change the value of `LogBufParallelism`, you also may need to change the value of `LogBufMB` to satisfy the constraint that `LogBufMB/LogBufParallelism` >=8.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `LogBufParallelism` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LogBufParallelism`	An integer value between 1 and 64. Default is 4.
Windows ODBC Data Source Administrator	**LogBufParallelism** field	An integer value between 1 and 64. Default is 4.

LogFileSize

The `LogFileSize` attribute specifies the maximum size of transaction log files in megabytes. The minimum value is 8 MB. The default value is 64 MB. If you specify a size smaller than 8 MB, TimesTen returns an error message. Before TimesTen release 11.2.1.4, the minimum size was 1MB. If you created your database in a previous release of TimesTen and specified a log file size of less than 8 MB, you must increase the value assigned to this attribute to avoid an error.

Actual transaction log file sizes may be slightly smaller or larger than `LogFileSize` because log records cannot span transaction log files.

A value of zero indicates that either the default transaction log file size should be used if the database does not exist, or that the transaction log file size in effect for the most recent connection should be used if the database does exist.

It is best to set the value of `LogFileSize` to match or exceed `LogBufMB`, although it is possible that the `LogBufMB` value can be greater than the `LogFileSize` value. The log buffer cannot be larger than the `LogFileSize` value, so if the buffer is not sized to match `LogFileSize`, the buffer capacity may not be fully utilized.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set LogFileSize as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LogFileSize`	*n* - Size of transaction log file in megabytes. Default is 64 when the database is created and 0 (current size in effect) on subsequent connections. The minimum size is 8 MB. The maximum value is 1024 MB.
Windows ODBC Data Source Administrator	**Log files Size (MB)** field	Size of transaction log file in megabytes. Default is 64 when the database is created and 0 (current size in effect) on subsequent connections. The minimum size is 8 MB. The maximum value is 1024 MB.

LogFlushMethod

Controls the method used by TimesTen to write and sync log data to transaction log files. The overall throughput of a system can be significantly affected by the value of this attribute, especially if the application chooses to commit most transactions durably.

As a general rule, use the value 2 if most of your transactions commit durably and use the value 1 otherwise.

For best results, however, experiment with both values using a typical workload for your application and platform. Although application performance may be affected by this attribute, transaction durability is not affected. Changing the value of this attribute does not affect transaction durability in any way.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `LogFlushMethod` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LogFlushMethod`	**0** - Write data to the transaction log files using the previously used value. **1** - Write data to transaction log files using buffered writes and use explicit sync operations as needed to sync log data to disk (for example with durable commits). (This is the default). **2** - Write data to transaction log files using synchronous writes such that explicit sync operations are not needed.
Windows ODBC Data Source Administrator	**Log Flush Method** dropdown list	**0** - Write data to the transaction log files using the previously used value. **1** - Write data to transaction log files using buffered writes and use explicit sync operations as needed to sync log data to disk (for example with durable commits). This is the default. **2** - Write data to transaction log files using synchronous writes such that explicit sync operations are not needed.

See also

DurableCommits

Logging

Logging to disk enables applications to roll back transactions. Logging to disk incurs a performance penalty due to both operations needed to maintain the log and delays incurred in writing the log to disk. Logging to disk enables applications to roll back unwanted transactions.

By default, transaction logging is enabled. This is the only mode currently available in TimesTen.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `Logging` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`Logging`	**1** - Logs modifications to the database to disk (only value currently allowed).
Windows ODBC Data Source Administrator	Not available	Not available

LogPurge

If the `LogPurge` attribute is set, TimesTen automatically removes transaction log files when they have been written to both checkpoint files and there are no transactions that still need the transaction log files' contents. The first time checkpoint is called, the contents of the transaction log files are written to one of the checkpoint files. When checkpoint is called the second time, TimesTen writes the contents of the transaction log files to the other checkpoint file.

TimesTen purges the transaction log files if all these conditions are met:

- The contents of the transaction log files have been written to both checkpoint files.
- The transaction log files are not pending incremental backup.
- If replication is being used, the transaction log files have been replicated to all subscribers.
- If XLA is being used, all XLA bookmarks have advanced beyond the transaction log files.
- The transaction log files are not being used by any distributed transactions using the XA interface.

If this attribute is set to 0 or unchecked, unneeded transaction log files are appended with the `.arch` suffix. Applications can then delete the files.

This attribute is relevant only if `Logging` is set to 1. See also "MatchLogOpts" on page 1-56.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set LogPurge as follows:

Where to set the attributes	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LogPurge`	**0** - Does not remove old transaction log files at connect and checkpoint. **1** - Removes old transaction log files at connect and checkpoint (default).
Windows ODBC Data Source Administrator	**LogPurge** check box	**unchecked** - Does not remove old transaction log files at connect and checkpoint. **checked** - Removes old transaction log files at connect and checkpoint (default).

MemoryLock

On Solaris, Linux, Windows 64-bit and HP-UX 11 systems, TimesTen allows applications that connect to a shared database to specify whether the real memory should be locked while the database is being loaded into memory or while the store is in memory. If the physical memory used for the database is locked, the operating system's virtual memory sub-system cannot borrow that memory for other uses. No part of the database is ever paged out but this could lead to memory shortages in a system that is under configured with RAM. While memory locking can improve database load performance, it may impede other applications on the same computer.

On AIX the MemoryLock attribute is not implemented. The shared memory segment is locked when you use large pages, on AIX. You can lock the shared segment by using large pages. The Oracle TimesTen In-Memory Database Installation Guide contains more details about large pages.

The PL/SQL shared memory segment is not subject to `MemoryLock`.

Required privilege

Only the instance administrator can change the value of this attribute.

On Linux systems, set the *`groupname`* in the `memlock` setting to be the same as the instance administrator in the `/etc/security/limits.conf` file. Set the value of `memlock` to be at least as large as the TimesTen database shared memory segment.

On Solaris systems, the instance administrator must be `root` to set MemoryLock to 1 or 2. Setting MemoryLock to 3 or 4 enables use of Solaris "intimate shared memory".

On HP-UX systems, users need the MLOCK privilege, which is enabled with the `setprivgrp` command.

Setting

Set `MemoryLock` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`MemoryLock`	**0** - Does not lock memory (default). **1** - Tries to obtain a memory lock. If unable to lock, the connection succeeds. If a lock is obtained, it is released after the database is loaded into memory (recommended). **2** - A memory lock is required. If unable to lock, the connection fails. If a lock is obtained, the connection succeeds and the lock is released after the database is loaded into memory. **3** - Tries to obtain and keep a memory lock. If unable to lock, the connection succeeds. If a memory lock is obtained, the connection succeeds and the memory lock is held until the database is unloaded from memory. **4** - A memory lock is required and is held until the database is unloaded from memory. If unable to lock, the connection fails. If a lock is obtained, the connection succeeds and the memory lock is held until the database is unloaded from memory.
Windows ODBC Data Source Administrator	**Memory Lock** field	**0** - Does not lock memory (default). **1** - Tries to obtain a memory lock. If unable to lock, the connection succeeds. If a lock is obtained, it is released after the database is loaded into memory (recommended). **2** - A memory lock is required. If unable to lock, the connection fails. If a lock is obtained, the connection succeeds and the lock is released after the database is loaded into memory. **3** - Tries to obtain and keep a memory lock. If unable to lock, the connection succeeds. If a memory lock is obtained, the connection succeeds and the memory lock is held until the database is unloaded from memory. **4** - A memory lock is required and is held until the database is unloaded from memory. If unable to lock, the connection fails. If a lock is obtained, the connection succeeds and the memory lock is held until the database is unloaded from memory.

Overwrite

If the `Overwrite` attribute is set and there is an existing database with the same database path name as the new database, TimesTen destroys the existing database and creates a new empty database, as long as the existing database is not in use. If the `Overwrite` attribute is set and there is not a database with the specified database path name, TimesTen only creates a new database if the `AutoCreate` attribute is also set (see "AutoCreate" on page 1-23). `Overwrite` is ignored if `AutoCreate` is set to 0. Applications should use caution when specifying the `Overwrite` =1 attribute.

Required privilege

Only the instance administrator can change the value of this attribute. If a user other than an instance administrator attempts to connect to a database with `OVERWRITE`=1, TimesTen returns an error.

Setting

Set `Overwrite` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`Overwrite`	**0** - Does not overwrite existing database with the same path name (default). **1** - Overwrites existing database with the same path name.
Windows ODBC Data Source Administrator	Not available	Not available

PermSize

Indicates the size in MB of the permanent memory region for the database. You may increase `PermSize` at first connect but not decrease it. TimesTen returns a warning if you attempt to decrease the permanent memory region size. If the database does not exist, a `PermSize` value of 0 or no value indicates that the default size should be used. Default size is 32 MB. For an existing database, a value of 0 or no value indicates that the existing size should not be changed.

Once you have created a database, you can make the permanent partition larger, but not smaller. See "Specifying the size of a database" in *Oracle TimesTen In-Memory Database Operations Guide*.

The `ttMigrate` and `ttDestroy` utilities can also be used to change the Permanent Data Size, when appropriate.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `PermSize` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PermSize`	*n* - Size of permanent partition of the database, in megabytes; default is 32 MB for both 32-bit systems and 64-bit systems. Minimum size is 32 MB.
Windows ODBC Data Source Administrator	**Permanent Data Size** field	*n* - Size of permanent partition of the database, in megabytes; default is 32 MB for both 32-bit systems and 64-bit systems. Minimum size is 32 MB.

ReceiverThreads

This attribute controls the number of threads used to apply changes on the active master database to the standby master database in an active standby pair replication scheme. The default is 1. You can also set this attribute on one or more read-only subscribers in an active standby pair replication scheme to increase replication throughput from the standby master database to the subscribers.

By default, a receiver thread in the replication agent applies the changes to the standby master database. When this attribute is set to 2, an additional thread applies the changes. Databases must be hosted on systems that are 2-way or larger to take advantage of setting this attribute to 2.

If you set this attribute to 2 on the standby master database, you should also set it to 2 on the active master database to maintain increased throughput if there is a failover.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `ReceiverThreads` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`ReceiverThreads`	*n* - The number of threads used to apply changes from the active master database to the standby master database. You can also set this attribute on one or more read-only subscribers in an active standby pair replication scheme to increase replication throughput from the standby master database to the subscribers. The possible values are 1 and 2. Default is 1.
Windows ODBC Data Source Administrator	**ReceiverThreads** field	*n* - The number of threads used to apply changes from the active master database to the standby master database. You can also set this attribute on one or more read-only subscribers in an active standby pair replication scheme to increase replication throughput from the standby master database to the subscribers. The possible values are 1 and 2. Default is 1.

RecoveryThreads

The `RecoveryThreads` attribute determines the number of threads used to rebuild indexes during recovery.

If `RecoveryThreads=1`, during recovery, indexes that must be rebuilt are done serially. If you have enough processors available to work on index rebuilds on your computer, setting this attribute to a number greater than 1 can improve recovery performance. The performance improvement occurs only if different processors can work on different indexes. There is no parallelism in index rebuild within the same index.

The value of `RecoveryThreads` can be any value up to the number of CPUs available on your system.

The default is 1 when the database is created. Upon subsequent connections, if the database must be recovered and RecoveryThreads is unspecified or has a value of 0, then TimesTen uses the previous setting for this attribute.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `RecoveryThreads` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`RecoveryThreads`	*n* - The number of threads to use when rebuilding indexes during recovery. Default is 1 when the database is created and 0 on subsequent connections.
Windows ODBC Data Source Administrator	**RecoveryThreads** field	*n* - The number of threads to use when rebuilding indexes during recovery. Default is 1 when the database is created and 0 on subsequent connections.

Notes

For a progress report on the recovery process, see the rebuild messages in the support log.

Set the number of threads low enough to leave sufficient resources on the server for other services/processes.

TempSize

`TempSize` indicates the total amount of memory in MB allocated to the temporary region.

`TempSize` has no predefined value. If left unspecified, its value is determined from `PermSize` as follows:

- If `PermSize` is less than 64MB, `TempSize = 32 MB + ceiling(PermSize / 4 MB)`.
- Otherwise, `Tempsize = 40 MB + ceiling(PermSize / 8 MB)`.

TimesTen rounds the value up to the nearest MB.

If specified, TimesTen always honors the `TempSize` value. Since the temporary data partition is recreated each time a database is loaded, the `TempSize` attribute may be increased or decreased each time a database is loaded. For an existing database, a value of 0 or no value indicates that the existing size should not be changed. The minimum `TempSize` is 32 MB.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `TempSize` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`TempSize`	*n* - Size of temporary partition of the database, in megabytes. Minimum size is 32 MB on all platforms.
Windows ODBC Data Source Administrator	**Temporary Data Size** field	*n* - Size of temporary partition of the database, in megabytes. Minimum size is 32 MB on all platforms.

General connection attributes

General connection attributes are set by each connection and persist for the duration of the connection. General connection attributes are listed in Table 1–3, " General connection attributes" and described in detail in this section.

ConnectionName

This attribute is also available as a Client connection attribute.

This attribute allows you to attach a symbolic name to any database connection. Connection names are unique within a process.

The symbolic name is used to help identify the connection in various TimesTen administrative utilities, such as ttIsql, ttXactAdmin and ttStatus. This can be particularly useful with processes that make multiple connections to the database, as is typical with multithreaded applications or in the identification of remote clients.

The value of this attribute is intended to be dynamically defined at connection time using the connection string. The default value is the connecting executable file name. It can also be defined statically in the DSN definition. Values used for `ConnectionName` should follow SQL identifier syntax rules.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `ConnectionName` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`ConnectionName`	Enter a string up to 30 characters that represents the name of the connection.If the specified or default connection name is in use, TimesTen assigns the name con*n*, where *n* is an integer greater than 0 to make the name unique.If not specified, the connecting process name.
Windows ODBC Data Source Administrator	**Connection** field	Enter a string up to 30 characters that represents the name of the connection.If the specified or default connection name is in use, TimesTen assigns the name con*n*, where *n* is an integer greater than 0 to make the name unique.If not specified, the connecting process name.

DDLCommitBehavior

This attribute controls transactional commit behavior in relation to DDL (Data Definition Language) statements

You can set it to the traditional TimesTen behavior or to the Oracle database behavior.

- Traditionally, in TimesTen databases, DDL statements are executed as part of the current transaction and are committed or rolled back along with the rest of the transaction.
- The Oracle database issues an implicit `COMMIT` before and after any DDL statement.

> **Note:** If PLSQL support is enabled, the DDLCommitBehavior must be the Oracle transactional commit behavior (value 0).
>
> Do not use DDL statements in XA transactions.

DDL statements include:

- `CREATE`, `ALTER` or `DROP` of any database object (including tables, views, users procedures, indexes, etc.)
- `TRUNCATE`
- `GRANT` and `REVOKE`

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `DDLCommitBehavior` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`DDLCommitBehavior`	**0** - Oracle database style behavior. An implicit transaction commit is done before the execution of the DDL statement and a durable commit is done after execution of DDL statements. (default). **1** - Traditional TimesTen style behavior. Execution of DDL statements does not trigger implicit transaction commits.
Windows ODBC Data Source Administrator	**DDLCommitBehavior** field	**0** - Oracle database style behavior. An implicit transaction commit is done before the execution of the DDL statement and a durable commit is done after execution of DDL statements. (default). **1** - Traditional TimesTen style behavior. Execution of DDL statements does not trigger implicit transaction commits.

Examples

Example 1–1 TimesTen commit behavior

```
AUTOCOMMIT OFF;
CREATE TABLE t1 (c1 Varchar2(10));
COMMIT;

INSERT INTO t1 VALUES('some data');
1 row inserted.

CREATE TABLE t2 (c1 INTEGER);

ROLLBACK;

SELECT * FROM t1;
0 rows found.

SELECT * FROM t2;
2206: Table ttuser.t2 not found
The command failed.

INSERT INTO t1 VALUES('more data');
1 row inserted.

CREATE TABLE t1 (c1 VARCHAR2(10));
 2207: Table t1 already exists
The command failed.

ROLLBACK;

SELECT * FROM t1;
0 rows found.
```

Example 1–2 Oracle commit behavior

This example shows Oracle behavior (`DDLCommitBehavior`=0). In this example, the `INSERTs` and the creation of table t2 are committed. The second insert ('more data') is committed even though the DDL statement triggering the commit (duplicate create of table t1) fails:

```
-- implicit commit here

Command> CREATE TABLE t1 (c1 varchar2(10));
Table created.

-- implicit commit here

Command> COMMIT;
Commit complete.

Command> INSERT INTO t1 VALUES('some data');
1 row created.

-- implicit commit here
Command> CREATE TABLE t2 (c1 INTEGER);
Table created.

-- implicit commit here
```

```
SQL> ROLLBACK;
Rollback complete.

Command> SELECT * FROM t1;
C1
----------
some data

Command> SELECT * FROM t2;
no rows selected

Command> INSERT INTO t1 VALUES('more data');
1 row created.
--

implicit commit here

Command> CREATE TABLE t1 (c1 VARCHAR2(10));
CREATE TABLE t1 (c1 VARCHAR2(10))
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object

-- implicit rollback

Command> ROLLBACK;
Rollback complete.

Command> SELECT * FROM t1;
C1
----------
some data
more data
```

Diagnostics

Allows an application to configure the level of diagnostics information generated by TimesTen for the connection. TimesTen diagnostics messages are warnings whose numbers lie within the range 20000 through 29999. `Diagnostics` connection attribute values are integers.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set Diagnostics as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	Diagnostics	**0** - No diagnostics messages are generated. **1** - Base level diagnostics messages are generated. (default).
Windows ODBC Data Source Administrator	**Diagnostics** field	**0** - No diagnostics messages are generated. **1** - Base level diagnostics messages are generated. (default).

DuplicateBindMode

This attribute determines whether applications use traditional TimesTen parameter binding for duplicate occurrences of a parameter in a SQL statement or Oracle-style parameter binding.

Traditionally, in TimesTen, multiple instances of the same parameter name in a SQL statement are considered to be multiple occurrences of the *same* parameter. When assigning parameter numbers to parameters, TimesTen assigns parameter numbers only to the first occurrence of each parameter name. The second and subsequent occurrences of a given name do not get their own parameter numbers. In this case, A TimesTen application binds a value for every unique parameter in a SQL statement. It cannot bind different values for different occurrences of the same parameter name nor can it leave any parameters or parameter occurrences unbound.

In Oracle Database, multiple instances of the same parameter name in a SQL statement are considered to be different parameters. When assigning parameter numbers, Oracle assigns a number to each parameter occurrence without regard to name duplication. An Oracle application, at a minimum, binds a value for the first occurrence of each parameter name. For the subsequent occurrences of a given parameter, the application can either leave the parameter occurrence unbound or it can bind a different value for the occurrence.

For more details on parameter binding, see *Oracle TimesTen In-Memory Database SQL Reference*.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `DuplicateBindMode` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`DuplicateBindMode`	**0** - Use the Oracle parameter binding model. (default) **1** - Use the traditional TimesTen parameter binding model.
Windows ODBC Data Source Administrator	**Duplicate Bind Mode** check box	**unchecked** - Use the Oracle parameter binding model. (default) **checked** - Use the traditional TimesTen parameter binding model.

Notes

When using Oracle Call Interface, DuplicateBindMode must be set to 0.

When PLSQL is set to 1 and DuplicateBindMode is set to 1, PL/SQL programs may not issue SQL statements containing duplicate parameter names.

DurableCommits

By default, DurableCommits is set to 0. This means that a log record is written to the file system when a transaction is committed, but the log record is not immediately written to disk. This reduces transaction execution time at the risk of losing some committed transactions if a failure occurs. When DurableCommits is set to 1, a log record is written to disk when the transaction is committed.

A connection can also call the ttDurableCommit built-in procedure to do durable commits explicitly on selected transactions. A call to ttDurableCommit flushes the log buffer to disk. The log buffer is shared among all connections and contains log records from transactions of all connections.

Log records are continually copied from the file system to disk. You can use LogFlushMethod to control when the file system is synchronized with the disk.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set DurableCommits as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX ODBC.INI file	DurableCommits	**0** - Does not force log to disk on transaction commit (default). **1** - Forces log to disk on transaction commit.
Windows ODBC Data Source Administrator	**Durable Commits** check box	**unchecked** - Does not force log to disk on transaction commit. **checked** - Forces log to disk on transaction commit

See also

LogFlushMethod

Isolation

By default, TimesTen uses read committed isolation. The Isolation attribute specifies the initial transaction isolation level for the connection. For a description of the isolation levels, see "Concurrency control through isolation and locking" in *Oracle TimesTen In-Memory Database Operations Guide*.

The value may be modified by an `ALTER SESSION` statement. For details, see *Oracle TimesTen In-Memory Database SQL Reference*.

If the passthrough or the propagate IMDB Cache feature is used, the TimesTen isolation level setting is inherited by the Oracle session. TimesTen serializable mode is mapped to Oracle's serializable mode. TimesTen read committed mode is mapped to Oracle's read committed mode. For more details on the passthrough attribute+9, see "PassThrough" on page 1-96.

With PassThrough set to 3, you must use an `ALTER SESSION` statement to permanently modify the isolation level on the Oracle connection. For example on a connection to the DSN repdb1_1121:

1. Call `ttIsql` and connect to the `DSN` with passthrough level 3:

```
% ttsisql;
Command> connect "dsn=repdb1_1121;passtrhough=3";
Connection successful:. . .PassThrough=3; TypeMode=0;
<default setting Autocommit=1>
```

2. Turn off AutoCommit:

```
Command> autocommit=0;
```

3. Temporarily change the PassThrough level to 0:

```
Command> passthrough=0;
```

4. Alter the isolation level to serializable:

```
Command> prepare 1 ALTER SESSION SET ISOLATION_LEVEL=serializable;
Command> commit;
Command> exec=1;
```

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `Isolation` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`Isolation`	**0** - Connects to database in serializable isolation mode. **1** - Connects to database in read committed mode (default).

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	**Isolation** dropdown list	**0** - Connects to database in serializable isolation mode. **1** - Connects to database in read committed isolation mode (default).

LockLevel

By default, TimesTen enables row-level locking for maximum concurrency. With row-level locking, transactions usually obtain locks on the individual rows that they access, although a transaction may obtain a lock on an entire table if TimesTen determines that doing so would result in better performance. Row-level locking is the best choice for most applications, as it provides the finest granularity of concurrency control. To use row-level locking, applications must set the LockLevel connection attribute to 0 (the default value). To cache Oracle tables, you must set row-level locking. In order to `CREATE`, `DROP`, or `ALTER` a user, you can only use row-level locking and thus, the Locklevel must be set to 0 before you can perform any of these operations.

To give every transaction in this connection exclusive access to the database, you can enable database-level locking by setting the LockLevel attribute to 1. Doing so may improve performance for some applications.

A connection can change the desired lock level at any time by calling the `ttLockLevel` built-in procedure. Connections can also wait for unavailable locks by calling the `ttLockWait` built-in procedure. Different connections can coexist with different levels of locking, but the presence of even one connection doing database-level locking leads to loss of concurrency. To display a list of all locks on a particular database you can use the `ttXactAdmin` utility.

When using PL/SQL in your applications, set `LockLevel=0` and selectively change to database level locking for specific transactions that require that level of locking by using the `ttLockLevel` built-in procedure.

Required privilege

Setting `LockLevel` to 1 requires `ADMIN` privilege.

Setting

Set `LockLevel` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LockLevel`	**0** - Transactions access database using row-level locking (default). **1** - Transactions access database by acquiring an exclusive lock on the entire database.
Windows ODBC Data Source Administrator	**DS-Level Locking** check box	**unchecked** - Transactions access database using row-level locking (default). **checked** - Transactions access database by acquiring an exclusive lock on the entire database.

LockWait

Allows an application to configure the lock wait interval for the connection. The lock wait interval is the number of seconds to wait for a lock when there is contention on it. Sub-second `LockWait` values significant to tenths of a second can be specified using decimal format for the number of seconds. For example:

```
LockWait = 0.1
```

results in a lock wait of one tenth of a second.

`LockWait` may be set to any value between 0 and 1,000,000 inclusive to a precision of tenths of a second. The default is 10 seconds:

```
LockWait = 10.0
```

Actual lock wait response time is imprecise and may be exceeded by up to one tenth of a second, due to the scheduling of the agent that detects timeouts. This imprecision does not apply to zero second timeouts, which are always reported immediately.

Cache grid uses message wait time with lock wait time. When using cache grid, lock wait times are approximately half the value you have specified. If your applications require the full lock wait time, specify twice the desired seconds.

A connection can change the lock wait interval at any time by calling the `ttLockWait` built-in procedure.

To display a list of all locks on a particular database you can use the TimesTen utility `ttXactAdmin`.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `LockWait` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`LockWait`	*s* - Seconds to wait for locking conflict resolution before timing out. Default is 10 seconds.
Windows ODBC Data Source Administrator	**LockWait** field	*s* - Seconds to wait for locking conflict resolution before timing out. Default is 10 seconds.

MatchLogOpts

The first connection to a database determines the type of logging that is performed and whether the transaction log files are purged. Any subsequent connection must specify the same values for the `Logging` and `LogPurge` attributes or TimesTen generates an error. If a connection does not know the current state of these attributes, `MatchLogOpts` can be set so that the logging attributes match.

> **Note:** If `MatchLogOpts` is set to True for the first connector, an error is generated and the connection fails. Because of this, use the attribute with caution.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `MatchLogOpts` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`MatchLogOpts`	**0** - Values of Logging and LogPurge are used (default). **1** - Values of Logging and LogPurge are ignored. Instead, values match those of current connections.
Windows ODBC Data Source Administrator	**Match Log Opts** check box	**unchecked** - Values of Logging and LogPurge are used (default). **checked** - Values of Logging and LogPurge are ignored. Instead, values match those of current connections.

PermWarnThreshold

Indicates the threshold percentage at which TimesTen issues out-of-memory warnings for the permanent partition of the database's memory. The database is considered no longer out of permanent memory if it falls 10% below this threshold. An application must call the built-in procedure `ttWarnOnLowMemory` to receive out-of-memory warnings. The threshold also applies to SNMP warnings. See "ttWarnOnLowMemory" on page 2-169 and "Diagnostics through SNMP Traps" in *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `PermWarnThreshold` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PermWarnThreshold`	*p* - Percentage at which warning should be issued. Default is 90%
Windows ODBC Data Source Administrator	**Low Memory Warning Thresholds for Permanent Data** field	*p* - Percentage at which warning should be issued. Default is 90%.

PrivateCommands

When multiple connections execute the same command, they access common command structures controlled by a single command lock. To avoid sharing their commands and possibly placing contention on the lock, you can use `PrivateCommands`. This gives you better scaling at the cost of increased temporary space usage.

By default, the `PrivateCommands` is turned off and commands are shared.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `PrivateCommands` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PrivateCommands`	**0** - Commands are shared with other connections. (Default) **1** - Commands are not shared with any other connection.
Windows ODBC Data Source Administrator	**Private Commands** field	**0** - Commands are shared with other connections. (Default) **1** - Commands are not shared with any other connection.

Notes

If there are many copies of the same command, all of them are invalidated by a DDL or statistics change. This means that reprepare of these multiple copies takes longer when `PrivateCommands` = 1. With more commands DDL execution can take slightly longer.

When using the `PrivateCommands` attribute, memory consumption can increase considerably if the attribute is not used cautiously. For example, if `PrivateCommands`=1 for an application that has 100 connections with 100 commands, there are 10,000 commands in the system: one private command for each connection.

PWDCrypt

The PWDCrypt contains an encrypted version of the corresponding PWD value. The value for PWD is stored in clear text, which does not allow special characters, in the .odbc.ini file on UNIX and in the Windows Registry on Windows. Any users who have access to the .odbc.ini file or Windows Registry can view the value for this attribute. The PWDCrypt attribute allows special characters, is case sensitive and contains the value of the encrypted password.

For security reasons, the PWDCrypt attribute should only be placed in User DSNs or user private ODBCINI files. The presence of the PWDCrypt in System DSNs allows any user to use the PWDCrypt value to connect to TimesTen, even though they have no knowledge of the cleartext password.

To generate the value for this attribute, run the ttuser utility.

Required privilege

No privilege is required to change the value of this attribute.

Notes

If PWD and PWDCrypt are both supplied, the PWD value is used. See "UID and PWD" on page 1-64.

The PWD is not stored anywhere in the TimesTen system.

Setting

Set PWDCrypt as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX ODBC.INI file	PWDCrypt	Enter the value generated by the ttuser utility.
Windows ODBC Data Source Administrator	**PWDCrypt** field	Enter the value generated by the ttuser utility.

QueryThreshold

Use this attribute to write a warning to the support log and throw an SNMP trap when the execution time of a SQL statement exceeds the specified value. For queries executed by the replication agent, see *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*. You cannot set a query threshold for a SQL statement that is executed by the cache agent. The value of `QueryThreshold` applies to all connections. It applies to all SQL statements except those executed by the replication agent or the cache agent.

The value of this attribute can be any integer equal to or greater than 0. The default value is 0. A value of 0 indicates that no warning is issued. The unit is seconds.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `QueryThreshold` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`QueryThreshold`	A non-negative integer. Default is 0 and indicates that TimesTen does not return a warning.
Windows ODBC Data Source Administrator	**QueryThreshold (secs)** field	A non-negative integer. Default is 0 and indicates that TimesTen does not return a warning.

ReplicationTrack

Assigns a connection to a replication track. All transactions issued by the connection are assigned to this track, unless the track is altered.

To start user-specified parallel replication you must set a value for the `ReplicationParallelism` attribute, specifying the number of replication tracks to be applied in parallel. You must also set `ReplicationApplyOrdering` to 1.

The `TTREP.REPPEERS` system table `Track_ID` column shows the track associated with the connection.

You can use the `ALTER SESSION SQL` statement to assign or change the value of this attribute within a session.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `ReplicationTrack` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`ReplicationTrack`	*n* - An integer between 1 and 64 that specifies the replication track to be used by transactions issued by the connection.
Windows ODBC Data Source Administrator	**Replication Track** field	*n* - An integer between 1 and 64 that specifies the replication track to be used by transactions issued by the connection.

SQLQueryTimeout

Use this attribute to specify the time limit in seconds within which the database should execute SQL statements.

The value of this attribute can be any integer equal to or greater than 0. The default value is 0. A value of 0 indicates that the query does not time out.

This attribute does not stop IMDB Cache operations that are being processed on Oracle. This includes passthrough statements, flushing, manual loading, manual refreshing, synchronous writethrough, and propagating.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `SQLQueryTimeout` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`SQLQueryTimeout`	*n* - Time limit in seconds for which the database should execute SQL queries.
Windows ODBC Data Source Administrator	**QueryTimeout (secs)** field	*n* - Time limit in seconds for which the database should execute SQL queries.

TempWarnThreshold

Indicates the threshold percentage at which TimesTen issues out-of- memory warnings for the temporary partition of the database's memory. The database is considered no longer out of temporary memory if it falls 10% below this threshold. An application must call the built-in procedure `ttWarnOnLowMemory` to receive out-of-memory warnings. The threshold also applies to SNMP warnings. See "ttWarnOnLowMemory" on page 2-169 and "Diagnostics through SNMP Traps" in *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `TempWarnThreshold` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`TempWarnThreshold`	*p* - Percentage at which warning should be issued. Default is 90%.
Windows ODBC Data Source Administrator	**Low Memory Warning Thresholds for Temporary Data** field	*p* - Percentage at which warning should be issued. Default is 90%

UID and PWD

A user ID and password must be provided by a user who is identified internally to TimesTen. Alternatively, an encrypted password can be supplied using the `PWDCrypt` attribute. Some TimesTen operations prompt for the `UID` and `PWD` of the user performing the operation.

For client/server applications, specify `UID` and `PWD` either in the Client DSN configuration or in the connection string. The `UID` and `PWD` values specified in a connection string take precedence over the values specified in the Client DSN configuration.

When caching Oracle tables, `PWD` specifies the TimesTen password while `OraclePWD` specifies the Oracle password.

Required privilege

No privilege is required to change the values of these attributes.

Setting

Set `UID` and `PWD` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`UID`	Character string specifying the user ID.
C or Java programs or UNIX `ODBC.INI` file	`PWD`	Character string specifying the password that corresponds to the user ID.
Windows ODBC Data Source Administrator	**User ID** field	Character string specifying the user ID.

WaitForConnect

When an application requests a connection to a TimesTen database and the connection is not possible (perhaps during concurrent loading/recovery of a database), TimesTen normally waits for completion of the conflicting connection. In some cases, it can take some time for an application to connect to a database. If the `WaitForConnect` attribute is off and the database is not immediately accessible, TimesTen returns immediately an error. For a description of the error, look for the error message number in "Warnings and Errors" in *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `WaitForConnect` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`WaitForConnect`	**0** - Does not wait if connection to database fails. **1** - Waits until connection to database is possible (default).
Windows ODBC Data Source Administrator	**Wait For Connect** check box	**unchecked** - Does not wait if connection to database fails. **checked** - Waits until connection to database is possible (default).

NLS general connection attributes

NLS connection attributes are set by each connection and persist for the duration of the connection. These attributes control the globalization behaviors of the database. NLS general connection attributes are listed Table 1–4, " NLS general connection attributes" and described in detail in this section.

You can use the `ALTER SESSION` statement to change NLS parameters to override the values that are assigned to these attributes at connection time.

ConnectionCharacterSet

This attribute is also available as a Client connection attribute.

This attribute specifies the character encoding for the connection, which may be different from the database character set. This can be useful when you have multiple connections to a database and one or more of those connections requires a character set that differs from that specified in the database.

The connection character set determines the character set in which data is displayed or presented.

Generally, you should choose a connection character set that matches your terminal settings or data source. Your database character set should be chosen based on the data requirements. For example: Do you have data in Unicode or is your data in Japanese on UNIX (`EUC`) or Windows (`SJIS`)?

When the database and connection character sets differ, TimesTen performs data conversion internally based on the connection character set. If the connection and database character sets are the same, TimesTen does not need to convert or interpret the data set. Best performance occurs when connection and database character sets match, since no conversion is required.

Parameters and SQL query text sent to the connect should be in the connection character set. Results and error messages returned by the connection are returned in the connection character set.

Character set conversions are not supported for the `TIMESTEN8` character set. A `ConnectionCharacterSet` value of `TIMESTEN8` results in an error if the value assigned to the `DatabaseCharacterSet` is not `TIMESTEN8`.

This attribute accepts the same values used for the `DatabaseCharacterSet`. For a list of character set names that can be used as a value for this attribute, see "Supported Character Sets."

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `ConnectionCharacterSet` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`ConnectionCharacterSet`	The default value for `ConnectionCharacterSet` is `US7ASCII`, except when the database character set is `TIMESTEN8`.
Windows ODBC Data Source Administrator	**Connection CharacterSet** list	The default value for `ConnectionCharacterSet` is `US7ASCII`, except when the database character set is `TIMESTEN8`.

NLS_LENGTH_SEMANTICS

The NLS_LENGTH_SEMANTICS attribute is used to set the default length semantics configuration. Length semantics determines how the length of a character string is determined. The length can be treated as a sequence of characters or a sequence of bytes.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set NLS_LENGTH_SEMANTICS as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX ODBC.INI file	NLS_LENGTH_ SEMANTICS	Specify either BYTE (default) or CHAR
Windows ODBC Data Source Administrator	**NLS_LENGTH_ SEMANTICS** list	Select either BYTE (default) or CHAR.

NLS_NCHAR_CONV_EXCP

The NLS_NCHAR_CONV_EXCP attribute determines whether an error is reported when there is data loss during an implicit or explicit character type conversion between NCHAR/NVARCHAR2 data and CHAR/VARCHAR2 data. A replacement character is substituted for characters that cannot be converted.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set NLS_NCHAR_CONV_EXCP as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX ODBC.INI file	NLS_NCHAR_CONV_EXCP	0 - Errors are not reported when there is a data loss during character type conversion. (default) 1 - Errors are reported when there is a data loss during character type conversion.
Windows ODBC Data Source Administrator	**NLS_NCHAR_ CONV_EXCP** checkbox	checked - Error messages are not reported when there is a data loss during character type conversion. (default) unchecked - Error messages are reported when there is a data loss during character type conversion.

NLS_SORT

The NLS_SORT attribute indicates which collating sequence to use for linguistic comparisons. It accepts the values listed in "Supported Linguistic Sorts." All these values can be modified to do case-insensitive sorts by appending _CI to the value. To perform accent-insensitive and case-insensitive sorts, append _AI to the value.

For materialized views and cache groups, TimesTen recommends that you explicitly specify the collating sequence using the NLSSORT SQL function rather than using this attribute in the connection string or DSN definition.

NLS_SORT may affect many operations. The supported operations that are sensitive to collating sequence are:

- MIN, MAX
- BETWEEN
- =, <>, >, >=, <, <=
- DISTINCT
- CASE
- GROUP BY
- HAVING
- ORDER BY
- IN
- LIKE

Only BINARY sort is supported with the TIMESTEN8 character set.

NLS_SORT settings other than BINARY may have significant performance impact on character operations.

> **Note:** Primary key indexes are always based on the BINARY collating sequence. Use of non-BINARY NLS_SORT equality searches cannot use the primary key index

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set NLS_SORT as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX ODBC.INI file	NLS_SORT	Specify the linguistic sort sequence or BINARY (default).
Windows ODBC Data Source Administrator	**NLS_SORT** dropdown list	Specify the linguistic sort sequence or BINARY (default).

Supported linguistic sorts

The tables in this section list the supported values for the NLS_SORT general connection attribute and the NLS_SORT SQL function.

Monolingual linguistic sorts

Basic name	Extended name
ARABIC	-
ARABIC_MATCH	-
ARABIC_ABJ_SORT	-
ARABIC_ABJ_MATCH	-
ASCII7	-
AZERBAIJANI	XAZERBAIJANI
BENGALI	-
BIG5	-
BINARY	-
BULGARIAN	-
CANADIAN FRENCH	-
CATALAN	XCATALAN
CROATIAN	XCROATIAN
CZECH	XCZECH
CZECH_ PUNCTUATION	XCZECH_PUNCTUATION
DANISH	XDANISH
DUTCH	XDUTCH
EBCDIC	-
EEC_EURO	-
EEC_EUROPA3	-
ESTONIAN	-
FINNISH	-
FRENCH	XFRENCH
GERMAN	XGERMAN
GERMAN_DIN	XGERMAN_DIN
GBK	-
GREEK	-
HEBREW	-
HKSCS	-
HUNGARIAN	XHUNGARIAN
ICELANDIC	-
INDONESIAN	-
ITALIAN	-

Basic name	Extended name
LATIN	-
LATVIAN	-
LITHUANIAN	-
MALAY	-
NORWEGIAN	-
POLISH	-
PUNCTUATION	XPUNCTUATION
ROMANIAN	-
RUSSIAN	-
SLOVAK	XSLOVAK
SLOVENIAN	XSLOVENIAN
SPANISH	XSPANISH
SWEDISH	-
SWISS	XSWISS
THAI_DICTIONARY	-
TURKISH	XTURKISH
UKRAINIAN	-
UNICODE_BINARY	-
VIETNAMESE	-
WEST_EUROPEAN	XWEST_EUROPEAN

Multilingual linguistic sorts

Sort name	Description
CANADIAN_M	Canadian French sort supports reverse secondary, special expanding characters.
DANISH_M	Danish sort supports sorting uppercase characters before lowercase characters.
FRENCH_M	French sort supports reverse sort for secondary.
GENERIC_M	Generic sorting order which is based on ISO14651 and Unicode canonical equivalence rules but excluding compatible equivalence rules.
JAPANESE_M	Japanese sort supports SJIS character set order and EUC characters which are not included in SJIS.
KOREAN_M	Korean sort Hangul characters are based on Unicode binary order. Hanja characters based on pronunciation order. All Hangul characters are before Hanja characters.
SPANISH_M	Traditional Spanish sort supports special contracting characters.
THAI_M	Thai sort supports swap characters for some vowels and consonants.
SCHINESE_RADICAL_M	Simplified Chinese sort is based on radical as primary order and number of strokes order as secondary order.

Sort name	Description
SCHINESE_STROKE_M	Simplified Chinese sort uses number of strokes as primary order and radical as secondary order.
SCHINESE_PINYIN_M	Simplified Chinese Pinyin sorting order.
TCHINESE_RADICAL_M	Traditional Chinese sort based on radical as primary order and number of strokes order as secondary order.
TCHINESE_STROKE_M	Traditional Chinese sort uses number of strokes as primary order and radical as secondary order. It supports supplementary characters.

PL/SQL first connection attributes

PL/SQL connection attributes are set by each connection and persist for the duration of the connection. These attributes control the behaviors of the database. PL/SQL first connection attributes are listed Table 1–5, " PL/SQL first connection attributes" and described in detail in this section.

PLSQL

This attribute determines whether PL/SQL is configured for the database.

Specifying PLSQL=1 enables PL/SQL use in the database. Specifying PLSQL=0 disables PL/SQL use in the database.

On platforms where PL/SQL is supported, and in TimesTen installations where PL/SQL support was enabled at installation time, the default is PLSQL=1. In other environments the default is PLSQL=0.

PL/SQL may be enabled when the database is initially created, or at any first connect afterwards. However, once PL/SQL support is enabled in a database, you cannot disable it later.

Configuring PL/SQL support in a database results in the creation of several "built-in" PL/SQL packages and procedures that are defined in *Oracle TimesTen In-Memory Database PL/SQL Developer's Guide*.

Some things to be aware of when setting this attribute are:

- If an application connects to a database that has PL/SQL enabled, and the application or odbc.ini file specifies PLSQL=0, TimesTen returns a warning.
- If an application connects to a database that does not have PL/SQL enabled, and the application or odbc.ini file specifies PLSQL=1, what happens depends on whether this is a first connect or a subsequent one. At first connect, the database is configured to support PL/SQL. Otherwise, TimesTen returns an error.
- If PLSQL=0, all PL/SQL first and general connection attributes are ignored.
- If PLSQL=0, any attempt to change the value of a PL/SQL general connection attributes returns an error.
- If PLSQL=1, Logging must be set to 1.
- If PLSQL=1, we recommend setting LockLevel=0 for the connection. If database level locking is enabled, some PL/SQL internal functions cannot be performed. You can use the ttLockLevel built-in procedure to selectively change to database-level locking only for those specific transactions that require it.
- If PL/SQL support is enabled, the DDLCommitBehavior must be the Oracle transactional commit behavior (value 0).

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set PLSQL as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX ODBC.INI file	PLSQL	**0** - Indicates that PL/SQL is not enabled for the database. **1** - Enables PL/SQL for the database. (default)

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	**PL/SQL Enabled** checkbox	**checked** - Enables PL/SQL for the database. unchecked - Indicates that PL/SQL is not enabled for the database.

PLSQL_MEMORY_ADDRESS

This attribute determines the virtual address at which this shared memory segment is loaded into each process that uses the TimesTen "direct" drivers. This memory address must be identical in each process using TimesTen. You must specify the value as a hexadecimal address.

Use of PL/SQL requires a shared memory segment. This shared memory contains recently-executed PL/SQL code, shared package state, and metadata associated with the operation of PL/SQL. This shared memory segment is separate from the one containing the TimesTen database.

If PL/SQL use is enabled (PLSQL=1) and you have not specified a value for `PLSQL_MEMORY_ADDRESS`, TimesTen uses a platform-dependent default value.

The default values for each platform are designed to:

1) Maximize the amount of virtual space for your TimesTen database and for your applications.

2) Minimize the fragmentation of the virtual address space.

3) Avoid conflicts with other uses of virtual address apace.

The platform specific default memory addresses are:

Operating system	Address
32-bit Linux	`10000000`
64-bit Linux	`0000007fa0000000`
32-bit AIX	`c0000000`
64-bit AIX	`06ffffff00000000`
32-bit Solaris	`10000000`
64-bit Solaris	`ffffff0000000000`
32-bit Windows	`5B8C0000`
HP-UX systems	Do not specify a value. The operating system automatically manages the address of the memory segment.

Some things to consider when setting this attribute are:

- If applications simultaneously connect to multiple TimesTen databases in direct mode, then each database must use a different value for value for `PLSQL_MEMORY_ADDRESS`.
- The value of this attribute is stored persistently by TimesTen. This allows the attribute value to be properly specified in situations when the database is loaded automatically by TimesTen. For example, the database is automatically loaded if RamPolicy for the database is set to 1.
- If the PL/SQL shared memory cannot be mapped at the appropriate address, TimesTen returns an error and the connection to the database fails.
- The memory segment size is determined by the value of `PLSQL_MEMORY_ADDRESS`.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `PLSQL_MEMORY_ADDRESS` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PLSQL_MEMORY_ADDRESS`	A hexidecimal value that indicates the memory address for PL/SQL process.
Windows ODBC Data Source Administrator	**PL/SQL Memory Address** field	A hexidecimal value that indicates the memory address for PL/SQL process.

PLSQL_MEMORY_SIZE

Use of PL/SQL requires a shared memory segment. This attribute determines the size in megabytes of the shared memory segment used by PL/SQL, which is shared by all connections.

This shared memory contains recently-executed PL/SQL code, the shared package state, and metadata associated with the operation of PL/SQL. This shared memory segment is separate from the one containing the TimesTen database.

Some things to consider when setting this attribute are:

- The value of this attribute is stored persistently by TimesTen. This allows the attribute value to be properly specified in situations when the database is loaded automatically by TimesTen. For example, the database is automatically loaded if RamPolicy for the database is set to 1.
- The default memory size is 32 MB on UNIX systems and 23 MB on Windows 32-bit systems. The minimum size is 2MB, if PLSQL=1. For most PL/SQL users, the default memory size should be an adequate amount of memory. For databases that make extensive use of PL/SQL, specify a larger memory size. If the memory space is exhausted, `ORA-4031` errors may occur during PL/SQL execution.
- The address of the memory segment is determined by the value of `PLSQL_MEMORY_ADDRESS`.

For more information on how to calculate the size for `PLSQL_MEMORY_SIZE`, see "Calculate shared memory size for PL/SQL runtime" in *Oracle TimesTen In-Memory Database Operations Guide*

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `PLSQL_MEMORY_SIZE` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PLSQL_MEMORY_SIZE`	Specify a positive integer greater than 2 representing the size of the shared memory segment in megabytes. The default size is 32 MB.
Windows ODBC Data Source Administrator	**PL/SQL Memory Size** field	Specify a positive integer greater than 2 representing the size of the shared memory segment in megabytes. The default size is 23 MB.

PL/SQL general connection attributes

PL/SQL general connection attributes are set by each connection and persist for the duration of the connection. These attributes control the behaviors of the database. PL/SQL general connection attributes are listed Table 1–6, " PL/SQL general connection attributes" and described in detail in this section.

For more details on this attribute, see the appropriate Oracle 11g documentation.

You can use the `ALTER SESSION` statement to change PL/SQL parameters to override the values that are assigned to the PL/SQL general connection attributes at connection time. For details, see *Oracle TimesTen In-Memory Database SQL Reference*.

PLSCOPE_SETTINGS

PLSCOPE_SETTINGS controls whether the PL/SQL compiler generates cross-reference information. Either all or no cross-references are generated.

Some things to consider when setting this attribute are:

- The PLSCOPE_SETTINGS connection attribute determines the initial value of this attribute within a session. The value may be modified by an ALTER SESSION statement. If the attribute is specified in an ALTER SESSION statement in a database where PLSQL=0, an error is returned.
- If this attribute is specified in a connection string or in the odbc.ini file and the application is connecting to a database where PLSQL=0, no error or warning results.

Note: For more details on this attribute, see *Oracle Database PL/SQL Language Reference.*

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set PLSCOPE_SETTINGS as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX ODBC.INI file	PLSCOPE_SETTINGS	IDENTIFIERS:NONE (default) IDENTIFIERS:ALL
Windows ODBC Data Source Administrator	**PLScope settings** pulldown list	IDENTIFIERS:NONE (default) IDENTIFIERS:ALL

PLSQL_CCFLAGS

This attribute can be used to set directives to control conditional compilation of PL/SQL units, which allows you to customize the functionality of a PL/SQL program depending on conditions that are checked. This is especially useful when applications may be deployed to multiple database environments. Possible uses include activating debugging or tracing features, or basing functionality on the version of the database.

Use this format:

```
PLSQL_CCFLAGS = 'v1:c1,v2:c2,...,vn:cn'
```

vi has the form of an unquoted PL/SQL identifier. It is unrestricted and can be a reserved word or a keyword. The text is insensitive to case. Each one is known as a flag or flag name. Each *vi* can occur multiple times in the string, each occurrence can have a different flag value, and the flag values can be of different kinds.

ci is one of the following: a PL/SQL boolean literal, a `PLS_INTEGER` literal, or the literal `NULL`. The text is insensitive to case. Each one is known as a flag value and corresponds to a flag name.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `PLSQL_CCFLAGS` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PLSQL_CCFLAGS`	'A string literal with this format: `'v1:c1,v2:c2,...,vn:cn'` Default: null
Windows ODBC Data Source Administrator	**PL/SQL CCFlags** field	'A string literal with this format: `'v1:c1,v2:c2,...,vn:cn'` Default: null

You can use the ALTER SESSION SQL statement to change this attribute within a session.

PLSQL_CONN_MEM_LIMIT

This attribute specifies the *maximum* amount of process heap memory in megabytes that PL/SQL can use for the connection in which it is set.

Some things to consider when setting this attribute are:

- PL/SQL does not allocate this memory until or unless it is needed. Many PL/SQL programs require only a small amount of memory. How you write your application can determine memory requirements. For example, using large `VARRAYs` in PL/SQL code can require a lot of memory.
- If you attempt to allocate more memory than allowed, TimesTen returns an error.
- The value can be modified with the `ALTER SESSION` statement.

See *Oracle TimesTen In-Memory Database PL/SQL Developer's Guide* for more information.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `PLSQL_CONN_MEM_LIMIT` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PLSQL_CONN_MEM_LIMIT`	An integer value in MB. Default value is 100MB.
Windows ODBC Data Source Administrator	**PL/SQL Connection Memory Limit** field	An integer value in MB. Default value is 100MB.

PLSQL_OPTIMIZE_LEVEL

This attribute specifies the optimization level to be used to compile PL/SQL library units. The higher the setting of this parameter, the more effort the compiler makes to optimize PL/SQL library units.

Some things to consider when setting this attribute are:

- The `PLSQL_OPTIMIZE_LEVEL` connection attribute determines the initial value of this attribute within a session. The value may be modified by an `ALTER SESSION` statement. If the attribute is specified in an `ALTER SESSION` statement in a database where `PLSQL=0`, an error is returned.
- If this attribute is specified in a connection string or in the `odbc.ini` file and the application is connecting to a database where `PLSQL=0`, no error or warning results.

Note: For more details on this attribute, see *Oracle Database PL/SQL Language Reference.*

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `PLSQL_OPTIMIZE_LEVEL` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PLSQL_OPTIMIZE_LEVEL`	For details on the settings for this attribute, see *Oracle Database PL/SQL Language Reference.* The default value is 2.
Windows ODBC Data Source Administrator	**PL/SQL Optimization Level** pulldown list	For details on the settings for this attribute, see *Oracle Database PL/SQL Language Reference.* The default value is 2.

PLSQL_TIMEOUT

This attribute controls how long (in seconds) PL/SQL program units, including PL/SQL procedures, anonymous blocks and functions, are allowed to run before being automatically terminated.

This value may be modified with an `ALTER SESSION` statement. If this value is modified through `ALTER SESSION`, the new value impacts any PL/SQL program units that are currently running.

> **Note:** The frequency with which PL/SQL programs check execution time against this timeout value is variable. It is possible for programs to run significantly longer than the timeout value before being terminated.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `PLSQL_TIMEOUT` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PLSQL_TIMEOUT`	A positive integer representing the number of seconds for the timeout value. A value of 0 means that there is no timeout limit. The default value is 30 seconds.
Windows ODBC Data Source Administrator	**PL/SQL Timeout** field	A positive integer representing the number of seconds for the timeout value. A value of 0 means that there is no timeout limit. The default value is 30 seconds.

IMDB Cache first connection attributes

IMDB Cache first connection attributes are used only when you are using the IMDB Cache product. IMDB Cache first connection attributes are listed in Table 1–7, " IMDB Cache first connection attributes" and described in detail in this section.

CacheAWTMethod

Determines whether PL/SQL execution method or SQL array execution method is used for Asynchronous Writethrough propagation to apply changes to the Oracle server.

By default, Asynchronous Writethrough (AWT) uses SQL array execution to apply changes within TimesTen to the Oracle database. This method, `CacheAWTMethod=0`, works well when the same type of operation is repeated. For example, array execution is very efficient when a user does an update that affects several rows of the table. Updates are grouped together and sent to the Oracle server in one batch.

If you specify the PL/SQL execution method, `CacheAWTMethod=1`, AWT bundles all pending operations into a single PL/SQL collection that is sent to the Oracle server to be executed. This method can improve AWT throughput when there are mixed transactions and network latency between TimesTen and the Oracle server.

PL/SQL execution method transparently falls back to array execution mode temporarily when it encounters one of the following:

- A statement that is over 32761 bytes in length.
- A statement that references a column of type `BINARY FLOAT`, `BINARY DOUBLE` and `VARCHAR` of length greater than 4000 bytes.

Specify the SQL execution method, `CacheAWTMethod=0`, if any AWT cache group contains a `VARBINARY` column.

The `SYSTEMSTATS` table contains information about the number of times the execution method temporarily falls back to SQL array execution.

> **Note:** Use the same AWT execution method on all TimesTen nodes in any active standby pair replication scheme.

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set `CacheAWTMethod` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`CacheAWTMethod`	**0** - Use SQL array execution method. (default) 1 - use PL/SQL collections and anonymous blocks (PL/SQL execution method).
Windows ODBC Data Source Administrator	**Cache AWT Method** field	**0** - Use SQL array execution method. (default) 1 - use PL/SQL collections and anonymous blocks (PL/SQL execution method).

IMDB Cache database attributes

IMDB Cache connection attributes are used only when you are using the IMDB Cache product. IMDB Cache data store attributes are listed in and described in detail in this section.

CacheGridEnable

Determines whether cache grid is enabled or disabled. The TimesTen database must be a member of a cache grid before you can create cache groups. The default is 1 (enabled).

Required privilege

Only the instance administrator can change the value of this attribute.

Setting

Set CacheGridEnable as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	CacheGridEnable	**0** - Cache groups can be defined outside of a cache grid. **1** - All cache groups in the database must be defined as members of a cache grid. (default)
Windows ODBC Data Source Administrator	**Cache Grid Enable** check box	**unchecked** - Cache groups can be defined outside of a cache grid. **checked** - All cache groups in the database must be defined as members of a cache grid. (default)

CacheGridMsgWait

Specifies the number of seconds that an application waits for a message response from a remote member in a cache grid.

For more information on caching Oracle data in a TimesTen cache grid, see *Oracle In-Memory Database Cache User's Guide*.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `CacheGridMsgWait` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`CacheGridMsgWait`	Set to the number of seconds that TimesTen should wait for a cache grid message from a remote member. The default is 60 seconds.
Windows ODBC Data Source Administrator	**Cache Grid Message Wait** field	Set to the number of seconds that TimesTen should wait for a cache grid message from a remote member. The default is 60 seconds.

IMDB Cache general connection attributes

IMDB Cache general connection attributes are used only when you are using the IMDB Cache product. IMDB Cache general connection attributes are listed in Table 1–9, " IMDB Cache general connection attributes" and described in detail in this section.

DynamicLoadEnable

This attribute enables or disables dynamic load of Oracle data to a TimesTen dynamic cache group. By default, dynamic load of Oracle data is enabled.

To enable or disable dynamic load at the statement level and temporarily override the setting of this attribute, set the `DynamicLoadEnable` optimizer flag with the `ttOptSetFlag` built-in procedure.

> **Note:** The value of this attribute overrides the dynamic load behavior of all dynamic cache groups for the current connection to the database.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `DynamicLoadEnable` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`DynamicLoadEnable`	**0** - Disables dynamic load of Oracle data to TimesTen dynamic cache groups for the current connection. **1** - Enables dynamic load of Oracle data to TimesTen dynamic cache groups for the current connection. (default)
Windows ODBC Data Source Administrator	**Dynamic Load Enable** field	**0** - Disables dynamic load of Oracle data to TimesTen dynamic cache groups for the current connection. **1** - Enables dynamic load of Oracle data to TimesTen dynamic cache groups for the current connection. (default)

DynamicLoadErrorMode

This attribute controls what happens when an application executes a SQL operation against a dynamic cache group and the SQL operation cannot use dynamic load.

With a value of 0, the SQL operation executes against whatever data is in the TimesTen cache tables and returns a result based on that data with no error indicated.

With a value of 1, any statement that cannot use dynamic load (even if it does not need dynamic load) fails with an error indicating that it is not dynamic load-compliant.

For more information on caching Oracle data in a TimesTen cache group, see *Oracle In-Memory Database Cache User's Guide*.

Note: To override the value of this attribute at the statement level, set the `DynamicLoadErrorMode` optimizer flag with the ttOptGetFlag built-in procedure.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `DynamicLoadErrorMode` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`DynamicLoadErrorMode`	**0** - Statements execute against the cached data with no error. (default) **1** - Statements use dynamic load or fail with an error.
Windows ODBC Data Source Administrator	**DynamicLoadErrorMode** field	**0** - Statements execute against the cached data with no error. (default) **1** - Statements use dynamic load or fail with an error.

OracleNetServiceName

The `OracleNetServiceName` attribute is used with the IMDB Cache.

This attribute identifies the Service Name for the Oracle instance

To cache Oracle tables and enable communication with Oracle, you must specify an Oracle Service Name.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `OracleNetServiceName` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`OracleNetServiceName`	Character string specifying the Oracle Service Name that is to be used as the Oracle ID.
Windows ODBC Data Source Administrator	**OracleNetServiceName** field	Character string specifying the Oracle Service Name that is to be used as the Oracle ID.

OraclePWD

This attribute is used with IMDB Cache. It identifies the password for the user specified by `UID` to connect to the Oracle database to perform cache operations.

Required privilege

No privilege is required to set the value of this attribute.

Setting

This attribute must be set in the connection string. On Linux, suppose you have defined the following `odbc.ini` file:

```
[myDSN]
Datastore=/data/myDSN
PermSize=128
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
```

Set `OraclePWD` for user `ttuser` by connecting to `myDSN` as follows:

```
% ttisql

Copyright (c) 1996-2009, Oracle.  All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.


Command> connect "dsn=myDSN;OraclePWD=mypwd";
Connection successful:
DSN=beta4;UID=ttuser;DataStore=/data/myDSN;DatabaseCharacterSet=AL32UTF8;
ConnectionCharacterSet=AL32UTF8;PermSize=128;TypeMode=0;
(Default setting AutoCommit=1)
Command>
```

On Windows, set `OraclePWD` in the connection string in the same way that it is set on Linux.

See also

"UID and PWD" on page 1-64

PassThrough

This attribute is used with the IMDB Cache.

It specifies which SQL statements are executed only in the cache database and which SQL statements are passed through to the Oracle database. For more information about the IMDB Cache, see *Oracle In-Memory Database Cache User's Guide* and "CREATE CACHE GROUP" in *Oracle TimesTen In-Memory Database SQL Reference*.

The execution of a prepared PassThrough command assumes that the schema of dependent objects in the Oracle database has not changed since the prepare. If the schema has changed the PassThrough command may cause unexpected results from the Oracle database.

When passing SQL statements through to Oracle, use only TimesTen supported data types in column definitions. If the specified data type is not supported in TimesTen, the passthrough statement fails.

For information on changing the isolation level on the Oracle connection, when using this attribute, see "Isolation" on page 1-52.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `PassThrough` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`PassThrough`	**0** - SQL statements are executed only on TimesTen. (default) **1** - `INSERT`, `UPDATE` and `DELETE` statements are executed on TimesTen unless they reference one or more tables that are not in TimesTen. If they reference one or more tables not in TimesTen, they are passed through to the Oracle database. DDL statements are executed on TimesTen. Other statements are passed through to the Oracle database if they generate a syntax error in TimesTen or if one or more tables referenced within the statement are not in TimesTen. **2** - `INSERT`, `UPDATE` and `DELETE` statements performed on tables in read-only cache groups or user managed cache groups with the `READONLY` cache table attribute are passed through to the Oracle database. Passthrough behavior for other cache group types is the same as `PassThrough=1`. **3** - All statements are passed through to the Oracle database for execution, except that `INSERT`, `UPDATE` and `DELETE` statements issued on cache tables in a dynamic AWT global cache group result in a TimesTen error. **4** - `SELECT` statements issues on cache tables in a dynamic AWT global cache group that do not satisfy the criteria for a dynamic load query are passed through to the Oracle database for execution. Otherwise, statements are executed in the TimesTen database. **5** - `SELECT` statements issued on cache tables in a dynamic AWT global cache group that do not satisfy the criteria for a dynamic load query are passed through to the Oracle database for execution when all committed updates on cache tables in dynamic AWT global cache groups by previous transactions within the connection have been propagated to the Oracle database. Otherwise, statements are executed in the TimesTen database.

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	**PassThrough** List	**0** - SQL statements are executed only on TimesTen. (default) **1** - `INSERT`, `UPDATE` and `DELETE` statements are executed on TimesTen unless they reference one or more tables that are not in TimesTen. If they reference one or more tables not in TimesTen, they are passed through to the Oracle database. DDL statements are executed on TimesTen. Other statements are passed through to the Oracle database if they generate a syntax error in TimesTen or if one or more tables referenced within the statement are not in TimesTen. **2** - `INSERT`, `UPDATE` and `DELETE` statements performed on tables in read-only cache groups or user managed cache groups with the `READONLY` cache table attribute are passed through to the Oracle database. Passthrough behavior for other cache group types is the same as `PassThrough=1`. **3** - All statements are passed through to the Oracle database for execution, except that `INSERT`, `UPDATE` and `DELETE` statements issued on cache tables in a dynamic AWT global cache group result in a TimesTen error. **4** - `SELECT` statements issues on cache tables in a dynamic AWT global cache group that do not satisfy the criteria for a dynamic load query are passed through to the Oracle database for execution. Otherwise, statements are executed in the TimesTen database. **5** - `SELECT` statements issued on cache tables in a dynamic AWT global cache group that do not satisfy the criteria for a dynamic load query are passed through to the Oracle database for execution when all committed updates on cache tables in dynamic AWT global cache groups by previous transactions within the connection have been propagated to the Oracle database. Otherwise, statements are executed in the TimesTen database.

Certain restrictions must be considered when using the passthrough feature. They include:

- If the `PassThrough` attribute is set so that a query must be executed in the Oracle database, the query is sent to the Oracle database without any changes. If the query uses a synonym for a table in a cache group, then a synonym with the same name must be defined for the corresponding Oracle table for the query to be successful.

- In the case that a SQL statement that uses TimesTen only syntax is passed through to the Oracle database, TimesTen returns an error message that indicates the syntax is not supported in Oracle.
- Execution of a prepared passthrough command assumes that the schema of dependent objects in the Oracle database have not changed after the prepare. If the schema has changed, unexpected results can occur.
- TimesTen does not include a cache invalidation feature. TimesTen does not verify that the cached tables are up to date. When a query is syntactically correct in TimesTen and the cache contains all the tables referenced in the query, the query is executed in TimesTen regardless of whether the cache is up to date.
- The passthrough of Oracle `INSERT`, `UPDATE` and `DELETE` operations depends on the setting of the PassThrough attribute as described in the table above. IMDB Cache cannot detect `INSERT`, `UPDATE` and `DELETE` operations that are hidden in a trigger or stored procedure. Therefore, TimesTen cannot enforce the passthrough rule on hidden operations.
- You cannot pass PL/SQL blocks through to the Oracle database.
- The effects of a passthrough `INSERT`, `UPDATE` or `DELETE` operation on a read-only cache group are only seen after the transaction is committed and after the next autorefresh operation is completed.
- There is no mechanism to detect or block updates on an Oracle table that is cached in a TimesTen synchronous writethrough cache group. Whether the updates are made by statements passed through the cache or from other Oracle applications, the changes are never reflected in IMDB Cache.
- Oracle Call Interface (OCI) does not support a mechanism to describe the binding type of the input parameters. Ensure that your application supplies the correct SQL types for passthrough statements. The ODBC driver converts the C and SQL types and presents the converted data and the SQL type code to TimesTen. TimesTen presents the information to OCI. The length of the input binding values is restricted to 4000 for `LONG` and `LONG RAW` types.
- At all passthrough levels, if a transaction has passed any DDL or DML statements to the Oracle database, then commits and rollbacks are executed in both Oracle and TimesTen. Otherwise, they are executed only in TimesTen.

RACCallback

This attribute allows you to enable or disable the installation of Transparent Application Failover (TAF) and Fast Application Notification (FAN) callbacks when using Oracle Real Application Clusters (Oracle RAC) with IMDB Cache.

For more information about IMDB Cache, see *Oracle In-Memory Database Cache User's Guide* and "CREATE CACHE GROUP" in *Oracle TimesTen In-Memory Database SQL Reference*.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `RACCallback` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`RACCallback`	**0** - Do not install TAF and FAN callbacks. **1** -Install the TAF and FAN callbacks. (default)
Windows ODBC Data Source Administrator	**RACCallback** check box	**unchecked** - Do not install TAF and FAN callbacks. **checked** - Install the TAF and FAN callbacks. (default).

TimesTen Client connection attributes

TimesTen Client connection attributes are used only when you are connecting to a TimesTen server from a TimesTen client application. TimesTen Client connection attributes are listed in Table 1–10, " Client connection attributes" and described in detail in this section.

In addition to the attributes listed in this section, some database attributes and general connection attributes are also available for client connections or impact the behavior of the connection. These attributes are:

- `ConnectionCharacterSet`
- `ConnectionName`
- UID and PWD

TCP_Port

When connecting to a TimesTen database using the TimesTen Client and Server, the TimesTen Client requires the network address and the TCP port number of the computer running the TimesTen Server. As a convenience, TimesTen allows you to define a logical server name that contains the network address and port number pair.

If you specify anything other than a logical server name for the `TTC_Server` attribute in the Client DSN definition, TimesTen Client assumes that the Server is running on the default TCP/IP port number. In such cases, if your Server is running on a port other than the default port, you must specify the port number in the ODBC connection string. For example:

```
"TTC_SERVER=server_host_name;TTC_SERVER_DNS=Server_DSN;TCP_PORT=server_port"
```

or

```
"DSN=Client_DSN;TCP_Port=server_port"
```

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `TCP_Port` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs	`TCP_Port`	Specify the port number where the Server is listening.
Windows ODBC Data Source Administrator and UNIX `ODBC.INI` file	TimesTen does not support specifying this attribute directly in a UNIX `ODBC.INI` file or in the Windows ODBC Data Source Administrator. Alternatively, TCP_Port can be defined in the logical server name.	N/A

TCP_Port2

This attribute is used to specify the port number to use if an automatic failover occurs. See the description of `TCP_Port` for details on setting the value of this attribute and associated attributes.

See *Oracle TimesTen In-Memory Database Operations Guide* for more information on automatic client failover.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `TCP_Port2` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs	`TCP_Port2`	Specify the failover port number where the Server should listen.
Windows ODBC Data Source Administrator and UNIX `ODBC.INI` file	TimesTen does not support specifying this attribute directly in a UNIX `ODBC.INI` file or in the Windows ODBC Data Source Administrator. Alternatively, `TCP_Port` can be defined in the logical server name.	N/A

TTC_FailoverPortRange

Specifies a port range for the port that the automatic client failover thread listens on for failover notifications in an active/standby replication configuration. The failover configuration allows a client application to connect to a new active node automatically if there is a failure on the current node.

Specifying a port range helps accommodate firewalls between the client and server systems. By default, TimesTen uses a port chosen by the operating system.

> **Note:** Client failover is only supported when the client is part of an active/standby pair replication configurations.

See *Oracle TimesTen In-Memory Database Operations Guide* for more information on automatic client failover.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `TTC_FailoverPortRange` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`TTC_FailoverPortRange`	Specify a lower value and an upper value for the port numbers in the format <lowervalue>-<upper value>.
Windows ODBC Data Source Administrator	**Failover Port Range** field	Specify a lower value and an upper value for the port numbers in the format <lowervalue>-<upper value>.

TTC_Server

When connecting to a TimesTen database using the TimesTen Client and Server, the TimesTen Client requires the specification of the network address and TCP port number of the computer running the TimesTen Server. As a convenience, TimesTen allows you to define a logical server name that contains the network address and port number pair. If you specify anything other than a logical server name for this attribute, TimesTen Client assumes that the Server is running on the default TCP/IP port number. In such cases, if your Server is running on a port other than the default port, you must specify the port number in the ODBC connection string. For example:

```
"TTC_SERVER=server_host_name;TTC_SERVER_DNS=Server_DSN;TCP_PORT=server_port"
```

or

```
"DSN=Client_DSN;TCP_Port=server_port"
```

Once the logical server name is defined, it can be used as the value for the TTC_Server attribute in a Client DSN. Therefore, multiple Client DSNs referencing the same computer that is running the TimesTen Server can use the same logical server name for the value of the TTC_Server attribute instead of having to specify repeatedly the same network address and port number within each of the Client DSNs.

Note: TimesTen recommends that you specify a logical server name for the `TTC_Server` attribute. However, you can also specify a domain name server (DNS), host name or IP address for the `TTC_Server` attribute. If you do not use a logical server name and the TimesTen Server is listening on a non-default port number, you must provide the port number in the ODBC connection string. For example:

```
"TTC_SERVER=server_host_name;TTC_SERVER_DNS=Server_DSN;
TCP_PORT=server_port"
```

or

```
"DSN=Client_DSN;TCP_Port=server_port"
```

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `TTC_Server` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`TTC_Server`	Character string specifying the logical server.
Windows ODBC Data Source Administrator	**Server Name or Network Address** field	Character string specifying the logical server.

TTC_Server2

This attribute is used to specify the logical server name to use if an automatic failover occurs. See the description of `TTC_Server` for details on setting the value of this attribute and associated attributes.

The value of this attribute can be can be the same as the value specified for `TTC_Server` if it is a virtual IP address.

If the client has already failed over and has connected to `TTC_Server2` and the connection fails, it connects to `TTC_Server`. It alternately attempts to connect to `TTC_Server` and `TTC_Server2` until the `TTC_TIMEOUT` attribute expires.

> **Note:** Client failover is only supported when the client is part of an active/standby pair replication configurations.

See *Oracle TimesTen In-Memory Database Operations Guide* for more information on automatic client failover.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `TTC_Server2` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`TTC_Server2`	Character string specifying the logical server to be used if an automatic failover occurs.
Windows ODBC Data Source Administrator	**Server Name or Network Address 2** field	Character string specifying the logical server to be used if an automatic failover occurs.

TTC_Server_DSN

The `TTC_Server_DSN` attribute specifies a Server DSN on the computer running the TimesTen Server.

WINDOWS

On Windows, Server DSNs are the set of TimesTen System DSNs that use the TimesTen Data Manager driver. Use the ODBC Data Source Administrator to define Server DSNs.

UNIX

On UNIX, Server DSNs are defined in the `/var/TimesTen/` *`instance`*`/sys.odbc.ini` file. More details on this topic can be found in the platform-specific sections.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `TTC_Server_DSN` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`TTC_Server_DSN`	Character string specifying the DSN that resides on the Server.
Windows ODBC Data Source Administrator	**Server DSN** field	Character string specifying the DSN that resides on the Server.

TTC_Server_DSN2

This attribute is used to specify the Server DSN on the computer running the TimesTen Server. This is the Server DSN to be used if an automatic failover occurs. See the description of `TTC_Server_DSN` for details on setting the value of this attribute and associated attributes.

If a failover occurs, if the client cannot connect to `TTC_Server_DSN` or loses the connection to the DSN, it attempts to connect to `TTC_Server_DSN2`.

> **Note:** Client failover is only supported when the client is part of an active/standby pair replication configurations.

See *Oracle TimesTen In-Memory Database Operations Guide* for more information on automatic client failover.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set `TTC_Server_DSN2` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or `Java programs` or `UNIX ODBC`.INI file	`TTC_Server_DSN2`	Character string specifying the DSN that resides on the Server to be used if an automatic failover occurs.
Windows ODBC Data Source Administrator	**Server DSN2** field	Character string specifying the DSN that resides on the Server to be used if an automatic failover occurs.

TTC_Timeout

The TTC_Timeout attribute sets a maximum time limit, in seconds, for an operation (e.g., a connection, SQL query) that is completed by using the TimesTen Client and Server. The TTC_Timeout attribute also determines the maximum number of seconds a TimesTen Client application waits for the result from the corresponding TimesTen Server process before timing out.

A value of 0 indicates that client/server operations should not timeout. Setting of this attribute is optional. If this attribute is not set, the default timeout period is 60 seconds. The maximum timeout period is 99,999 seconds. Upon timeout, the operation is interrupted, the Client application receives a timeout error and the connection is terminated. For example, if the Client application is running long queries, you may want to increase the timeout interval.

For certain queries, the client application may also set the SQL_QUERY_TIMEOUT ODBC statement option. The TimesTen Client ODBC Driver requires that SQL_QUERY_TIMEOUT must be less than TTC_TIMEOUT, unless the network timeout is set to 0. In that case, the network operation does not timeout.

The query timeout can be set using the SQLSetConnectOption ODBC call before a connection is established to the database using either the SQLConnect or SQLDriverConnect ODBC calls. Alternatively, the query timeout can be set by calling either the SQLSetConnectOption or SQLSetStmtOption ODBC calls after a connection is established to the database.

When the query timeout is set before establishing a connection to the database, the client driver does not know the network timeout value at that point. Hence, later, at connect time, the client driver silently sets the query timeout to a value slightly smaller than the network timeout value if:

- The network timeout value is greater than 0; and
- The query timeout value was 0, or greater than or equal to the network timeout value.

When the query timeout is set after establishing a connection to the database, the client driver returns an error if the network timeout value is greater than 0, and the query timeout value greater than or equal to the network timeout value. The SQLState is set to S1000.

This attribute is not supported when shared memory is used for Client/Server inter-process communication. If set, TimesTen ignores the attribute.

Required privilege

No privilege is required to change the value of this attribute.

Setting

Set TTC_Timeout as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX ODBC.INI file	TTC_Timeout	A value between 0 and 99999 that represents the number of seconds that TimesTen Client waits for a connection before timing out. (The default value is 60.)

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	**Timeout Interval** field	A value between 0 and 99999 that represents the number of seconds that TimesTen Client waits for a connection before timing out. (The default value is 60.)

Server connection attributes

Server connection attributes are specified in the Server DSN only and are read at first connection. See "Defining Server DSNs" in *Oracle TimesTen In-Memory Database Operations Guide*. The attributes are used to set the number of connections to a TimesTen server, the number of servers for each DSN and the size of each connection to the server. These attributes allow you to specify multiple client connections to a single Server. By default, TimesTen creates only one connection to a Server per child process.

> **Note:** These attributes must be specified in the DSN. If these attributes are specified in a connection string, TimesTen ignores them and their values.

There are also TimesTen main daemon options that can be used to specify multiple Server connections. In the case that both the daemon options and these attributes have been specified, the value of the attributes takes precedence.

Server connection attributes are listed in Table 1–11, " Server connection attributes" and described in detail in this section.

MaxConnsPerServer

The `MaxConnsPerServer` attribute sets the maximum number of concurrent connections to the server which the DSN references.

If you want to support many connections to the Server, you must make sure that the per-process file descriptor limit for the `UID` that TimesTen is being run as is set to a value somewhat more than the number of concurrent child servers that are active. This is the number of anticipated concurrent client connections divided by `MaxConnsPerServer`.

The value of this attribute takes precedence over the setting of the value of the`-maxConnsPerDSN` option in the `ttendaemon.options` file. For details, see "Specifying multiple connections to the TimesTen Server" in *Oracle TimesTen In-Memory Database Operations Guide*.

For limits on the maximum number of connections to a TimesTen database, see the "System Limits" chapter in the *Oracle TimesTen In-Memory Database System Tables and Limits Reference*.

Required privilege

Only a user with operating system privileges on the system DSN in which this attribute is defined can change the value of this attribute to a value other than the one currently in effect.

Setting

Set `MaxConnsPerServer` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`MaxConnsPerServer`	A value between 1 and 2047. The default is 1.
Windows ODBC Data Source Administrator	**Maximum Connections Per Server Process** field	A value between 1 and 2047. The default is 1.

ServersPerDSN

The `ServersPerDSN` attribute specifies the number of DSNs that can connect to a Server DSN at any given time.

The value of this attribute is only meaningful if the value of MaxConnsPerServer is greater than one. If there is only one connection per Server, the child server uses the process' main stack.

This value of this attribute takes precedence over the setting of the value of the `-serversPerDSN` option in the `ttendaemon.options` file. For details, see "Specifying multiple connections to the TimesTen Server" in *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

Only a user with operating system privileges on the system DSN in which this attribute is defined can change the value of this attribute to a value other than the one currently in effect.

Setting

Set `ServersPerDSN` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `ODBC.INI` file	`ServersPerDSN`	A value between 1 and 2047. The default is 1.
Windows ODBC Data Source Administrator	**Server Processes Per DSN** field	A value between 1 and 2047. The default is 1.

ServerStackSize

The `ServerStackSize` attribute value determines the size of the stack on the Server for each connection. The value of this attribute is only meaningful if the value of `MaxConnsPerServer` is greater than one. If there is only one connection per Server, the child server uses the process' main stack. It is also platform-dependent, as defined in the setting below.

This value of this attribute takes precedence over the setting of the `-serverStackSize` option in the `ttendaemon.options` file. For details, see "Specifying multiple connections to the TimesTen Server" in *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

Only a user with operating system privileges on the system DSN in which this attribute is defined can change the value of this attribute to a value other than the one currently in effect.

Setting

Set `ServerStackSize` as follows:

Where to set the attribute	How the attribute is represented	Setting

C or Java programs or UNIX `ODBC.INI` file	`ServerStackSize`	Valid values depend on the platform. The default is 128KB for 32-bit platforms and 256KB for 64-bit platforms.
		If the `sysconf` call is available, the minimum is:
		`sysconf(_SC_THREAD_STACK_MIN)/ 1024`
		else 0
		If the `getrlimit` call is available, the maximum value is:
		`getrlimit(RLIMIT_STACK, &r1);`
		r1.rlim_cur /1024
		else 4096
		The default is 128KB for 32-bit platforms and 256KB for 64-bit platforms.

Windows ODBC Data Source Administrator	**Server Stack Size** field	Valid values depend on the platform. The default is 128KB for 32-bit platforms and 256KB for 64-bit platforms.
		If the `sysconf` call is available, the minimum is:
		`sysconf(_SC_THREAD_STACK_MIN)/ 1024`
		else 0
		If the `getrlimit` call is available, the maximum value is:
		`getrlimit(RLIMIT_STACK, &r1);`
		r1.rlim_cur /1024
		else 4096

2

Built-In Procedures

TimesTen built-in procedures extend standard ODBC and JDBC functionality. You can invoke these procedures using the ODBC or JDBC procedure call interface. The procedure takes the position of the SQL statement, as illustrated in the following examples.

The following ODBC call tells the optimizer that it should not generate temporary hash indexes when preparing commands:

```
SQLExecDirect (hstmt, (SQLCHAR*)
       "{CALL ttOptSetFlag ('TmpHash', 0)}", SQL_NTS);
```

This is the equivalent JDBC call:

```
CallableStatement cstmt = con.prepareCall
          ("{CALL ttOptSetFlag ('TmpHash', 0)}");
cstmt.execute();
```

TimesTen built-in procedures can also be called from PL/SQL using the `EXECUTE IMMEDIATE` statement with `CALL`, as illustrated in the following example.

For example, to call the built-in procedure `ttConfiguration`, create a PL/SQL record type and then `SELECT INTO` that record type. Because `ttConfiguration` returns multiple rows, use `BULK COLLECT`.

```
Command> DECLARE
       >   TYPE ttConfig_record IS RECORD
       >      (name varchar2(255), value varchar2 (255));
       >   TYPE ttConfig_table IS TABLE OF ttConfig_record;
       > v_ttConfigs ttConfig_table;
       > BEGIN
       >  EXECUTE IMMEDIATE  'CALL ttConfiguration'
       >   BULK COLLECT into v_ttConfigs;
       >  DBMS_OUTPUT.PUT_LINE ('Name: ' || v_ttConfigs(1).name
       >    || ' Value: ' || v_ttConfigs(1).value);
       > end;
       > /
Name: CacheGridEnable Value: 0

PL/SQL procedure successfully completed.
```

Note: String parameter values for built-in procedures must be single-quoted as indicated in these examples, unless the value is `NULL`.

ttAgingLRUConfig

Description

This procedure sets the Least Recently Used (LRU) aging attributes on all regular tables that have been defined with an LRU aging policy. For cache tables, the aging policy is defined on the root table but applies to all tables in the cache group. The aging policy is defined on tables when they are created or altered, using the `CREATE TABLE` or `ALTER TABLE` SQL statements.

The LRU aging feature helps applications maintain the usage size of the database under a specified threshold by removing the least recently used data.

Data is removed if the database space in-use exceeds the specified threshold values. For cache groups, aging is defined at the root table for the entire cache instance. LRU aging is not allowed for cache groups with `AUTOREFRESH`. For those cache groups, use time-based aging.

Required privilege

This procedure requires no privilege to query the current values. It requires the `ADMIN` privilege to change the current values.

Syntax

```
ttAgingLRUConfig(LowUsageThreshHold, HighUsageThreshHold, AgingCycle)
```

Parameters

`ttAgingLRUConfig` has these optional parameters:

Parameter	Type	Description
lowUsageThreshold	BINARY_FLOAT	Sets, displays or resets the low end of percentage of database `PermSize`, specified in decimals. The bottom of the threshold range in which LRU aging should be deactivated. Default is 80%.
highUsageThreshold	BINARY_FLOAT	Sets, displays or resets the high end of percentage of database `PermSize`, specified in decimals. The top of the threshold range in which LRU aging should be activated. Default is 90%.

Parameter	Type	Description
agingCycle	TT_INTEGER	Sets, displays or resets the number of minutes between aging cycles, specified in minutes. Default is 1 minute. If you use this procedure to change the aging cycle, the cycle is reset based on the time that this procedure is called. For example, if you call this procedure at 12:00 p.m. and specify a cycle of 15 minutes, aging occurs at 12:15, 12:30, 12:45, etc.

Result set

ttAgingLRUConfig returns these results:

Column	Type	Description
lowUsageThreshold	BINARY_FLOAT NOT NULL	The current setting for the low end of percentage of database PermSize, specified in decimals.
highUsageThreshold	BINARY_FLOAT NOT NULL	The current setting for the high end of percentage of database PermSize, specified in decimals.
agingCycle	TT_INTEGER NOT NULL	The current setting for the number of minutes between aging cycles, specified in minutes.

Examples

To set the aging threshold to a low of 75 percent and a high of 95 percent and the aging cycle to 5 minutes, use:

```
CALL ttAgingLRUConfig (.75, .90, 5);
<.7500000, .9000000, 5>
```

To display the current LRU aging policy for all tables that defined with an LRU aging policy, call ttAgingLRUConfig without any parameters:

```
Call ttAgingLRUConfig();
```

If the tables are defined with the default thresholds and aging cycle, the procedure returns:

```
<.8000000, .9000000, 1>
1 row found.
```

To change the low usage threshold to 60%, the aging cycle to 5 minutes and to retain the previous high usage threshold, use:

```
Call ttAgingLRUConfig (60,,5);
< .6000000, .9000000, 5 >
1 row found.
```

Notes

The values of this procedure are persistent, even across system failures.

If no parameters are supplied, this procedure only returns the current LRU aging attribute settings.

See also

`ttAgingScheduleNow`
Oracle In-Memory Database Cache User's Guide

ttAgingScheduleNow

Description

This procedure starts the aging process, regardless of the value of the aging cycle. The aging process begins right after the procedure is called unless there is an aging process in progress. In that case, the new aging process begins when the aging process that was in process at the time the built-in was called has completed.

Aging occurs only once when you call this procedure. This procedure does not change any aging attributes. The previous aging state is unchanged. For example, if aging state is OFF when you call ttAgingScheduleNow, the aging process starts. When aging is complete, if your aging state is OFF, aging does not continue. To continue aging, you must call ttAgingScheduleNow again or change the aging state to ON, in which case aging occurs next based on the value of the aging cycle.

For tables with aging ON, the aging cycle is reset to the time when ttAgingScheduleNow was called. For example, if you call this procedure at 12:00 p.m. and the aging cycle is 15 minutes, aging occurs immediately and again at 12:15, 12:30, 12:45, etc.

If used in an external scheduler, such as a `cron` job, or executed manually, this procedure starts the aging process at the time the procedure is executed, if there is no aging process in progress, or as soon as the current aging process has completed. In the case that you want aging to occur *only* when the external scheduler executes the ttAgingScheduleNow procedure or you call it manually, set the aging state to OFF.

Aging is performed by a background thread that wakes up every second to check if any work must be done. Calling `ttAgingScheduleNow` only guarantees that the aging thread works on the specified tables within the next second, at best. If the aging thread is working on a different table at the time the built-in procedure is called, it may take some time to reach the specified table. The rows are visible until the aging thread commits the delete.

Required privilege

This procedure requires the `DELETE` privilege on the table being aged, or the `DELETE` ANY TABLE privilege when you do not specify a table.

Syntax

```
ttAgingScheduleNow ('tblname')
```

Parameters

`ttAgingScheduleNow` has the parameter:

Parameter	Type	Description
tblname	`TT_CHAR (61)`	The name of the table on which to start the aging process. If *tblName* is omitted, the aging process is started on all tables defined with any aging policy. Using a synonym to specify a table name is not supported.

Result set

`ttAgingScheduleNow` returns no results.

Examples

To schedule aging on all tables, including tables defined with both LRU aging and time-based aging, call `ttAgingScheduleNow` without any parameter values:

```
CALL ttAgingScheduleNow ();
```

This examples creates the table `agingex` with time-based aging policy and the aging state set to `OFF`. `ttAgingScheduleNow` is called, using the `ttIsql` utility, to start the aging process once. Rows are deleted from the table. After `ttAgingScheduleNow` is called, the aging state remains `OFF`. To continue aging, alter the table and set the aging state to `OFF`.

```
Command> CREATE TABLE agingex (col1 TT_INTEGER PRIMARY KEY NOT NULL,
    ts TIMESTAMP NOT NULL)
    AGING USE ts LIFETIME 1 MINUTES CYCLE 30 MINUTES OFF;

Command> DESCRIBE agingex;

Table TTUSER.AGINGEX:
Columns:
  *COL1             TT_INTEGER NOT NULL
   TS               TIMESTAMP (6) NOT NULL
Aging use TS lifetime 1 minute cycle 30 minutes off
1 table found.
(primary key columns are indicated with *)

Command> INSERT INTO agingex VALUES (1, SYSDATE);
1 row inserted.

Command> INSERT INTO agingex VALUES (2, SYSDATE);
1 row inserted.

Command> SELECT * FROM agingex;
< 1, 2007-03-25 13:06:29.000000 >
< 2, 2007-03-25 13:06:42.000000 >
2 rows found.

Command> CALL ttAgingScheduleNow ('agingex');

Command> SELECT * FROM agingex;
0 rows found.
```

See also

`ttAgingLRUConfig`
Oracle In-Memory Database Cache User's Guide

ttApplicationContext

Description

This procedure sets application-defined context for the next update record (either an `UPDATE` or commit) in order to pass application specific data to XLA readers.

Required privilege

This procedure requires no privilege.

Syntax

```
ttApplicationContext (cmd)
```

Parameters

`ttApplicationContext` has the parameter:

Parameter	Type	Description
cmd	`VARBINARY(16384) NOT NULL`	Context information to be passed.

Result set

`ttApplicationContext` returns no results.

Example

```
CALL ttApplicationContext (0x123);
```

See also

"XLA Reference" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttBackupStatus

Description

This procedure returns a single row with information about the current or last backup of the database. If a backup is in progress, this information represents the current backup. If no backup is in progress, this information represents the last backup taken.

If no backup has been taken on the database since the last first-connect, the status field is 0 and the rest of the columns are `NULL`.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttBackupStatus ()
```

Parameters

`ttBackupStatus` has no parameters.

Result set

`ttBackupStatus` returns the results:

Column	Type	Description
status	`TT_INTEGER NOT NULL`	An INTEGER code representing the current progress of a backup or the completion status of the last backup. Values are: 0 - No backup has been taken on the database since the last first-connect. 1 - A backup is currently in progress. 2 - The last backup completed successfully. 3 - The last backup failed. In this case the error column contains the error code for the failure.
destination	`TT_INTEGER`	The type of backup taken. The value is NULL when no backup has been taken on the database. Value is one of: 0 - Backup is/was being written to a file. 1 - Backup is/was being written to a stream. 2 - Backup is/was taken on behalf of replication duplicate.

Column	Type	Description
backupType	TT_INTEGER	Backup type, either full or incremental. The value is NULL when no backup has been taken on the database. Value is one of: 0 - Incremental backup 1 - Full backup
startTime	TT_TIMESTAMP	Time when the backup was started. The value is NULL when no backup has been taken on the database.
endTime	TT_TIMESTAMP	Time when the backup completed. If NULL and *startTime* is non-NULL, a backup is currently in progress.
backupLFN	TT_INTEGER	The transaction log file number of the backup point. The value is NULL when no backup has been taken on the database.
backupLFO	TT_INTEGER	The transaction log file offset of the backup point. The value is NULL when no backup has been taken on the database.
error	TT_INTEGER	If a backup fails, this column indicates the reason for the failure. The value is one of the TimesTen error numbers. The value is NULL when no backup has been taken on the database.
processId	TT_INTEGER	The ID of the process or daemon performing the backup (if known).

Example

```
CALL ttBackupStatus ();
< 2, 2, 1, 2005-08-12 13:10:32.587557, 2005-08-12 13:10:33.193269, 1, 1531840, 0,
6968 >
1 row found.
```

Notes

Does not return information about previous backups, other than the current or last one.

Information returned is not persistent across database startup or shutdown.

ttBlockInfo

Description

This procedure provides information about perm blocks and the amount of block-level fragmentation in a database.

Required privilege

This procedure requires no privilege.

Syntax

```
ttBlockInfo()
```

Parameters

`ttBlockInfo` has no parameters.

Result set

`ttBlockInfo` returns the result set:

Column	Type	Description
TotalBlocks	TT_BIGINT NOT NULL	Total number of blocks in the database.
FreeBlocks	TT_BIGINT NOT NULL	Total number of free blocks in the database.
FreeBytes	TT_BIGINT NOT NULL	Total size of the free blocks.
LargestFree	TT_BIGINT NOT NULL	Size of the largest free block.

Examples

```
CALL ttBlockInfo();
< 288, 3, 128711700, 128698596 >
1 row found.
```

ttBookmark

Description

This procedure returns information about the TimesTen transaction log. Records in the transaction log are identified by pairs of integers:

- A transaction log file number
- An offset in that transaction log file

Transaction log file numbers correspond to the file system names given to transaction log files. For example, the transaction log file `SalesData.log29` has the transaction log file number `29`.

Three log records are identified in the result row of `ttBookmark`:

- The identity of the most recently written log record
- The identity of the log record most recently forced to the disk
- The replication bookmark. The replication bookmark is the oldest log record that represents an update not yet replicated to another system

Required privilege

This procedure requires no privilege.

Syntax

```
ttBookmark()
```

Parameters

`ttBookmark` has no parameters.

Result set

`ttBookmark` returns the result set:

Column	Type	Description
writeLFN	TT_INTEGER	Last written transaction log file
writeLFO	TT_INTEGER	Last written offset in transaction log file
forceLFN	TT_INTEGER	Last transaction log file forced to disk
forceLFO	TT_INTEGER	Offset of last transaction log file forced to disk
holdLFN	TT_INTEGER	Replication bookmark transaction log file
holdLFO	TT_INTEGER	Replication bookmark log offset

Example

```
CALL ttBookmark ();
```

ttCacheAutorefresh

Description

This procedure starts an immediate autorefresh on the set of cache groups that are associated by sharing the same autorefresh interval with the specified cache group. This set of associated cache groups would normally be refreshed together automatically. The effect on the autorefresh process is the same as that of adding a new cache group with the same refresh interval as that of the specified cache group. This procedure is useful if updates have occurred on the Oracle database and you would like to refresh them on the cache group before the next scheduled autorefresh.

If there is an existing transaction with locks on table objects that belong to the set of cache groups to be autorefreshed, this procedure returns an error without taking any action. This procedure establishes a condition that requires that you commit or rollback before you can perform other work in the session.

Required privilege

This procedure requires the `CACHE_MANAGER` or `ADMIN` privilege.

Syntax

```
ttCacheAutorefresh ('cacheGroupOwner', 'cacheGroupName', synchronous)
```

Parameters

`ttCacheAutorefresh` has the parameters:

Parameter	Type	Description
cacheGroupOwner	`VARCHAR2 (30)`	Name of the cache group owner.
cacheGroupName	`VARCHAR2 (30) NOT NULL`	Name of the cache group.
synchronous	`TT_INTEGER`	Species whether data is updated on synchronously or asynchronously. **0** or **NULL** - Asynchronous mode. The procedure returns immediately. **1** - Synchronous mode. The procedure returns after the refresh operation has completed on all associated cache groups.

Result set

`ttCacheAutorefresh` returns no results:

Example

This example autorefreshes the *testcache* cache group and all cache groups with the same autorefresh interval. The procedure returns synchronously.

```
Command> call ttcacheautorefresh('user1','testcache', 1);
```

Notes

The specified cache group autorefresh state must be ON. While, other associated cache groups can be in any state, they are not refreshed if they are not in the autorefresh ON state.

An autorefresh of the specified associated cache groups cannot be in progress.

You cannot call this procedure on the standby node of an active standby pair.

This procedure is available only for IMDB Cache.

ttCacheAutorefreshStatsGet

Description

This procedure returns information about the last ten autorefresh transactions on the specified cache group. This information is only available when the AUTOREFRESH state is `ON` or `PAUSED`, and the cache agent is running.

The information returned by this built-in procedure is reset whenever:

- The cache agent is restarted
- The state is set to `OFF` and then back to `ON` or `PAUSED`
- The cache group is dropped and recreated

Required privilege

This procedure requires no privilege.

Syntax

```
ttCacheAutorefreshStatsGet ('cacheGroupOwner', 'cacheGroupName')
```

Parameters

`ttCacheAutorefreshStatsGet` has the parameters:

Parameter	Type	Description
cacheGroupOwner	`VARCHAR2 (30)`	Name of the cache group owner.
cacheGroupName	`VARCHAR2 (30) NOT NULL`	Name of the cache group for which autorefresh information should be returned.

Result set

The `ttCacheAutorefreshStatsGet` built-in procedure returns only a subset of column information for a cache group with autorefresh mode `FULL`. A column value of 0 returns for information that is not available.

`ttCacheAutorefreshStatsGet` returns the results:

Column name	Column type	Description	Returned for full autorefresh
cgId	`TT_BIGINT`	The cache group ID.	Y
startTimestamp	`TT_TIMESTAMP`	Timestamp when autorefresh started for this interval. See Notes section.	Y

Column name	Column type	Description	Returned for full autorefresh
cacheAgentUpTime	TT_BIGINT	Number of cache agent clock ticks in milliseconds at the time the autorefresh transaction started for this interval. This value is cumulative and is reset when the cache agent process starts. See Notes section.	Y
autorefNumber	TT_BIGINT	Autorefresh number for a cache group indicates the number of times this cache group has been incrementally refreshed since the cache agent started. This number will be initialized to 0 when the cache agent is started.	Y
autorefDuration	TT_BIGINT	The number of milliseconds spent in this autorefresh transaction.	Y
autorefNumRows	TT_BIGINT	The number of rows autorefreshed in this autorefresh. This includes all rows, including those in the root table and the child tables. If there are cache groups with multiple tables, child table rows get updated multiple times. Therefore, the number of rows autorefreshed may be more than number of rows updated on Oracle.	N
numOracleBytes	TT_BIGINT	The number of bytes transferred from Oracle in this autorefresh transaction.	N
autorefNumRootTblRows	TT_BIGINT	The number of root table rows autorefreshed in this autorefresh transaction.	Y
autorefQueryExecDuration	TT_BIGINT	The duration in milliseconds that it takes for the autorefresh query to execute on Oracle.	N
autorefQueryFetchDuration	TT_BIGINT	The duration in milliseconds that it takes for the autorefresh query to fetch rows from Oracle.	N
autorefTtApplyDuration	TT_BIGINT	The duration in milliseconds that it takes for TimesTen to apply the autorefresh.	N

Column name	Column type	Description	Returned for full autorefresh
totalNumRows	TT_BIGINT	The total number of rows autorefreshed since the cache agent started. The total number of rows autorefreshed may not be the same as number of rows updated on Oracle. This is because of a delay in marking the log; some updates may get autorefreshed and counted multiple times.	N
totalNumOracleBytes	TT_BIGINT	The total number of bytes transferred from Oracle since the cache agent started.	N
totalNumRootTblRows	TT_BIGINT	The total number of root table rows autorefreshed since the cache agent started.	Y
totalDuration	TT_BIGINT	The total autorefresh duration in milliseconds since the cache agent started.	Y
status	VARCHAR2 (128)	A string description of the status of the current autorefresh. See Note section. Supported values for this field are: ■ Complete ■ inProgress ■ Failed	Y

Example

In this example, *testcache* is a READONLY cache group with one table and an incremental autorefresh interval of 10 seconds.

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

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

```
< 1164260, 2007-07-23 15:41:22.000000, 700120, 36, 2120, 10240, 50330, 10240, 50,
 200, 1870, 1142680, 7361050, 1142680, 69770, Complete >
< 1164260, 2007-07-23 15:41:12.000000, 690110, 35, 0, 0, 0, 0, 0, 0, 0, 1132440,
 7310720, 1132440, 67650, Complete >
10 rows found.
```

Notes

Most of the column values reported above are collected at the cache group level. For example, *autorefDuration* and *autorefNumRows* only include information for the specified cache group. Exceptions to this rule are column values *cacheAgentUpTime, startTimestamp* and *autorefreshStatus*. These values are reported at the autorefresh interval level.

StartTimestamp is taken at the beginning of the autorefresh for the autorefresh interval. A cache group enters the "in progress" state as soon as the autorefresh for the interval starts. It is not marked "complete" until the autorefresh for all cache groups in the interval are complete.

This procedure is available only for IMDB Cache.

ttCacheAwtMonitorConfig

Description

This procedure enables monitoring to determine the amount of time spent in each component of the workflow of an AWT cache group. To display the monitoring results, use the `ttRepAdmin` utility with the -awtmoninfo and -showstatus commands.

If the replication agent is restarted, monitoring is turned off.

Setting the monitoring state to `OFF` resets the internal counters of the monitoring tool.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheAWTMonitorConfig ('state', samplingRate)
```

Parameters

`ttCacheAWTMonitorConfig` has the parameters:

Parameter	Type	Description
state	`TT_CHAR(10)`	Enables and disables AWT monitoring. Its value can be `ON` or `OFF`. Default is `OFF`
samplingRate	`TT_INTEGER`	Positive integer that specifies the frequency with which the AWT workflow is sampled. If *samplingFactor* is set to `1`, every AWT operation is monitored. Greater values indicate less frequent sampling. The value recommended for accuracy and performance is `16`. If *state* is set to `ON`, the default for *samplingFactor* is 16. If state is set to `OFF`, the default for *samplingFactor* is `0`.

Result set

`ttCacheAWTMonitorConfig` returns the following result if you do not specify any parameters. It returns an error if the replication agent is not running or if an AWT cache group has not been created.

Column	Type	Description
State	`TTVARCHAR (10) NOT NULL`	Current state of AWT monitoring. The value can be `ON` or `OFF`.
AWTSamplingFactor	`TT_INTEGER NOT NULL`	Positive integer that specifies the frequency with which the AWT workflow is sampled.

Examples

Example 2–1

Retrieve the current state and sampling factor when monitoring is disabled.

```
Command> CALL ttCacheAwtMonitorConfig;
< OFF, 0 >
1 row found.
```

Example 2–2

Enable monitoring and set the sampling frequency to 16.

```
Command> CALL ttCacheAwtMonitorConfig ('ON', 16);
< ON, 16 >
1 row found.
```

Example 2–3

Disable monitoring.

```
Command> CALL ttCacheAwtMonitorConfig; ('OFF')
< OFF, 0 >
1 row found.
```

See also

"ttRepAdmin" on page 3-88

ttCacheAWTThresholdGet

Description

This procedure returns the current transaction log file threshold for databases that include AWT cache groups.

Required privilege

This procedure requires no privilege.

Syntax

```
ttCacheAWTThresholdGet()
```

Parameters

`ttCacheAWTThresholdGet` has no parameters.

Result set

`ttCacheAWTThresholdGet` returns the result:

Column	Type	Description
threshold	`TT_INTEGER NOT NULL`	The number of transaction log files for all AWT cache groups associated with the database. If 0, there is no set limit.

Example

```
CALL ttCacheAWTThresholdGet();
```

Note

This procedure is available only for IMDB Cache.

See also

`ttCacheAWTThresholdSet`

ttCacheAWTThresholdSet

Description

This procedure sets the threshold for the number of transaction log files that can accumulate before AWT is considered either terminated or too far behind to catch up. This setting applies to all subscribers to the database. When the threshold is exceeded, updates are no longer sent to Oracle. If no threshold is set then the default is zero.

Using this built-in procedure, the threshold can be set after an AWT cache group has been created.

This setting can be overwritten by a `CREATE REPLICATION` statement that resets the Log Failure Threshold for the database.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheAWTThresholdSet(threshold)
```

Parameters

`ttCacheAWTThresholdSet` has the parameter:

Parameter	Type	Description
threshold	`TT_INTEGER`	Specifies the number of transaction log files for all AWT cache groups associated with the database. If the threshold is `NULL`, the log failure threshold is set to zero.

Result set

`ttCacheAWTThresholdSet` returns no results.

Example

To set the threshold to allow 12 transaction log files to accumulate, use:

```
CALL ttCacheAWTThresholdSet(12);
```

Notes

This procedure is available for Oracle In-Memory Database Cache.

The user is responsible to recover when the threshold is exceeded.

See also

`ttCacheAWTThresholdGet`

ttCacheConfig

Description

For all cache groups that cache data from the same Oracle instance, this procedure specifies a timeout value and recovery policies in the case that the Oracle Server is unreachable and the cache agent or database is considered terminated.

The automatic refresh state of the database and cache groups can be determined from the procedure `ttCacheDbCgStatus`.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheConfig(Param, tblOwner, tblName, Value)
```

Parameters

`ttCacheConfig` has these parameters:

Parameter	Type	Description
Param	`VARCHAR2(50) NOT NULL`	Specifies the parameter to be set by *Value*: ▪ `AgentTimeout` - Number of seconds before a database is declared terminated if the cache agent cannot connect to the Oracle server. ▪ `DeadDbRecovery` - Specifies the type of autorefresh recovery when the cache agent restarts. ▪ `TblSpaceFullRecovery` - Specifies the action that TimesTen takes when the cached Oracle table is updated and the cache administration user's tablespace is full. ▪ `TblSpaceThreshold` - Specifies the cache administration user's tablespace usage warning threshold as a percentage.
tblOwner	VARCHAR2(30)	Specifies the owner of the cached Oracle table. This parameter is required if *Param* is set `TblspaceFullRecovery`. Do not specify *tblOwner* for other values of *Param*. A synonym cannot be used to specify a table name.

Parameter	Type	Description
tblName	VARCHAR2(30)	Specifies the name of the cached Oracle table. This parameter is required if *Param* is set TblspaceFullRecovery. Do not specify *tblOwner* for other values of *Param*. Using a synonym to specify a table name is not supported.
Value	VARCHAR2(200)	Specifies the value to be set for *Param*. ■ When *Param* is AgentTimeout, it specifies the number of seconds before a database is declared terminated if the cache agent cannot connect to the Oracle server. The default is 0, which means that the database is never declared terminated. ■ When *Param* is DeadDbRecovery, the value can be Normal or Manual. Normal specifies a full automatic refresh. Manual specifies that REFRESH CACHE GROUP statement must be issued. The default is Normal. ■ When *Param* is TblSpaceFullRecovery, the value can be Reload or None. Reload specifies that rows are deleted from the change log table and a full automatic refresh is performed. None specifies that an Oracle error is returned when the cached Oracle table is updated. The default is Reload. ■ When *Param* is TblSpaceThreshold, the value can be 0 to 100. The default is 0, which means no warning is returned regardless of the tablespace usage.

Result set

ttCacheConfig returns no results when it is used to set parameter values. When it is used to return parameter settings, it returns these results:

Column	Type	Value
Param	VARCHAR2(50)	Parameter name: ▪ AgentTimeout ▪ DeadDbRecovery ▪ TblSpaceFullRecovery ▪ TblSpaceThreshold
tblOwner	VARCHAR2(30)	Owner of the cached Oracle table.
tblName	VARCHAR2(30)	Name of the cached oracle table. Using a synonym to specify a table name is not supported.
Value	VARCHAR2(200)	Specifies the value set for *Param*. ▪ When *Param* is AgentTimeout, it specifies the number of seconds before a database is declared terminated if the cache agent cannot connect to the Oracle server. ▪ When *Param* is DeadDbRecovery, the value can be Normal or Manual. ▪ When *Param* is TblSpaceFullRecovery, the value can be Reload or None. ▪ When *Param* is TblSpaceThreshold, the value can be 0 to 100.

Examples

To set the cache agent timeout to 600 seconds (10 minutes), enter:

```
CALL ttCacheConfig('AgentTimeout',,,'600');
```

To determine the current cache agent timeout setting, enter:

```
CALL ttCacheConfig('AgentTimeout');
< AgentTimeout, <NULL>, <NULL>, 600 >
1 row found.
```

To set the recovery method to Manual for cache groups whose automatic refresh status is dead, enter:

```
CALL ttCacheconfig('DeadDbRecovery',,,'Manual');
```

Configure the IMDB cache to prevent an automatic full refresh and receive an Oracle error when there is an update on a cached Oracle table while the cache administration user's tablespace is full. The Oracle table is terry.customer.

```
CALL ttCacheConfig('TblSpaceFullRecovery','terry','customer','None');
```

To determine the current setting for TblSpaceFullRecovery on the terry.customer cached Oracle table, enter:

```
CALL ttCacheConfig('TblSpaceFullRecovery','terry','customer');
< TblSpaceFullRecovery, TERRY, CUSTOMER, none >
```

```
1 row found.
```

To configure a warning to be returned when the cache administration user's tablespace is 85 percent full and an update operation occurs on the cached Oracle table, enter:

```
CALL ttCacheConfig('TblSpaceThreshold',,,'85');
```

Note

This procedure is available only for IMDB Cache.

See also

ttCacheDbCgStatus
ttCachePolicyGet
ttCachePolicySet
ttCacheStart
ttCacheStop
ttCacheUidGet
ttCacheUidPwdSet
ttAdmin

ttCacheDbCgStatus

Description

This procedure returns the automatic refresh status of the database and the specified cache group. If you do not specify any values for the parameters, the procedure returns the automatic refresh status for the database.

Required privilege

This procedure requires no privilege.

Syntax

```
ttCacheDbCgStatus([cgowner, cgName])
```

Parameters

`ttCacheDbCgStatus` has these optional parameters:

Parameter	Type	Description
cgOwner	`VARCHAR2(30)`	Specifies the user name of the cache group owner.
cgName	`VARCHAR2(30)`	Specifies the cache group name.

Result set

`ttCacheDbCgStatus` returns the result:

Column	Type	Value
dbStatus	`VARCHAR2(20)`	Specifies the status of all the cache groups in the database with respect to autorefresh. The status is one of: ▪ `alive` - The database is active. The status of all cache groups is `ok`. The cache agent has been in contact with the Oracle Server. ▪ `dead` - The cache agent was not able to contact Oracle within the timeout period. The status of all the cache groups with the autorefresh attribute is `terminated`. ▪ `recovering` - Some or all the cache groups with the autorefresh attribute are being resynchronized with the Oracle Server. The status of at least one cache group is `recovering`.

Column	Type	Value
cgStatus	VARCHAR2(20)	Specifies the autorefresh status of the specified cache group. The status is one of: ▪ ok - The specified cache group is synchronized with the Oracle database. The cache agent has been in contact with the Oracle Server. ▪ dead - The cache agent was not able to contact Oracle within the timeout period and the specified cache group may be out of sync with the Oracle Server. ▪ recovering - The specified cache group is being resynchronized with the Oracle Server.

Examples

This example shows that the automatic refresh status of the database is `alive`. The automatic refresh status of the cache group is `ok`.

```
CALL ttCacheDbCgStatus ('terry', 'cgemployees');
< alive, ok >
1 row found.
```

To determine the automatic refresh status of the database, call `ttCacheDbCgStatus` with no parameters:

```
CALL ttCacheDbCgStatus;
< dead, <NULL> >
1 row found.
```

Note

This procedure is available only for IMDB Cache.

See also

ttCacheConfig
ttCachePolicyGet
ttCachePolicySet
ttCacheStart
ttCacheStop
ttCacheUidGet
ttCacheUidPwdSet
ttAdmin

ttCacheDDLTrackingConfig

This procedure enables or disables tracking of DDL statements issued on cached Oracle tables. By default, DDL statements are not tracked.

DDL tracking saves the change history for all the cached Oracle tables. The SQL statement and when it was executed are written to a table in the cache administration user schema on Oracle. One DDL tracking table is created to store DDL statements issued on any cached Oracle table. This information can be used to diagnose autorefresh problems.

See "Tracking DDL statements issued on cached Oracle tables" in *Oracle In-Memory Database Cache User's Guide*.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheDDLTrackingConfig('trackingStatus')
```

Parameters

`ttCacheDDLTrackingConfig` has the parameter:

Parameter	Type	Description
trackingStatus	`TT_VARCHAR(10)`	Specifies whether DDL statements issued on cached Oracle tables are tracked. Valid values are: ▪ `enable` - Enables tracking ▪ `disable` - Disables tracking (default)

Result set

`ttCacheDDLTrackingConfig` returns no results.

Examples

```
Command> CALL ttCacheDDLTrackingConfig('enable');
```

ttCachePolicyGet

Description

This procedure returns the current policy used to determine when the TimesTen cache agent for the connected database should run. The policy can be either `always` or `manual`.

Required privilege

This procedure requires no privilege.

Syntax

```
ttCachePolicyGet()
```

Parameters

`ttCachePolicyGet` has no parameters.

Result set

`ttCachePolicyGet` returns the result:

Column	Type	Value
cachePolicy	`TT_VARCHAR(10)`	Specifies the policy used to determine when the TimesTen cache agent for the database should run. Valid values are: `always` - specifies that the agent for the database is always running. This option immediately starts the TimesTen cache agent. When the TimesTen daemon restarts, TimesTen automatically restarts the cache agent `manual` - (the default) specifies that you must manually start the cache agent using either the ttCacheStart built-in procedure or the ttAdmin `-cacheStart` command. You must explicitly stop the cache agent using either the ttCacheStop built-in procedure or the ttAdmin `-cacheStop` command.

Examples

To get the current policy for the TimesTen cache agent, use:

```
CALL ttCachePolicyGet ();
```

Note

This procedure is available only for IMDB Cache.

See also

`ttCacheConfig`
`ttCacheDbCgStatus`
`ttCachePolicySet`
`ttCacheStart`
`ttCacheStop`

ttCacheUidGet
ttCacheUidPwdSet
ttAdmin

ttCachePolicySet

Description

The procedure defines the policy used to determine when the TimesTen cache agent for the connected database should run. The policy can be either `always` or `manual`.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCachePolicySet('cachePolicy')
```

Parameters

`ttCachePolicySet` has these parameters:

Parameter	Type	Description
cachePolicy	`TT_VARCHAR(10) NOT NULL`	Specifies the policy used to determine when the TimesTen cache agent for the database should run. Valid values are: `always` - specifies that the agent for the database is always running. This option immediately starts the TimesTen cache agent. When the TimesTen daemon restarts, TimesTen automatically restarts the cache agent `manual` - (the default) specifies that you must manually start the cache agent using either the `ttCacheStart` built-in procedure or the `ttAdmin -cacheStart` command. You must explicitly stop the cache agent using either the `ttCacheStop` built-in procedure or the `ttAdmin -cacheStop` command. `norestart` - Specifies that the cache agent for the database is not to be restarted after a failure.

Result set

`ttCachePolicySet` returns no results.

Examples

To set the policy for TimesTen cache agent to `always`, use:

```
CALL ttCachePolicySet ('always');
```

Notes

This procedure is available only for IMDB Cache.

If you attempt to start the TimesTen cache agent (by changing the policy from `manual` to `always`) for a database with a relative path, TimesTen looks for the database relative to where TimesTen Data Manager is running, and fails. For example, on Windows, if you specify the path for the database as `DataStore=./payroll` and

attempt to start the TimesTen cache agent with this built-in procedure, the agent is not started because TimesTen Data Manager looks for the database in the *install_dir*\srv directory. On UNIX, TimesTen Data Manager looks in the /var/TimesTen/*instance* directory.

Successfully setting the policy to always automatically starts the cache agent if it was stopped.

See also

ttCacheConfig
ttCacheDbCgStatus
ttCachePolicyGet
ttCacheStart
ttCacheStop
ttCacheUidGet
ttCacheUidPwdSet
ttAdmin

ttCachePropagateFlagSet

Description

This procedure allows you to temporarily stop any updates from propagating to Oracle.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCachePropagateFlagSet(CommitsOn)
```

Parameters

`ttCachePropagateFlagSet` has the parameter:

Parameter	Type	Description
CommitsOn	`TT_INTEGER NOT NULL`	If 0, sets a flag to stop updates from being sent to Oracle. The flag remains set until the end of the transaction or until the procedure is set to 1. If 1, updates are sent to Oracle.

Result set

`ttCachePropagateFlagSet` returns no results.

Notes

This procedure is available only for IMDB Cache.

When using this procedure, it is important to turn off AutoCommit, otherwise after the procedure is called the transaction ends and propagation to Oracle is turned back on.

The propagate flag is reset after a commit or rollback.

If the value of `ttCachePropagateFlagSet` is re-enabled several times during a single transaction, the transaction is only partially propagated to Oracle.

`ttCachePropagateFlagSet` is the only built-in procedure that can be used in the same transaction as any of the other cache group operations, such as FLUSH, LOAD, REFRESH and UNLOAD.

ttCacheSqlGet

Description

This procedure generates the Oracle SQL statements to install or uninstall Oracle objects for:

- Read-only cache groups
- User managed cache groups with incremental autorefresh
- Asynchronous writethrough (AWT) cache groups

This is useful when the user creating the cache group does not have adequate privilege to write on the Oracle database. The Oracle DBA can then use the script generated by this built-in procedure to create the Oracle objects.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheSqlGet('feature_name', 'cache_group_name', install_flag)
```

Parameters

`ttCacheSqlGet` has these parameters:

Parameter	Type	Description
feature_name	`TT_VARCHAR (100)`	Can be specified as INCREMENTAL_ AUTOREFRESH or ASYNCHRONOUS_ WRITETHROUGH.
cache_group_name	`TT_VARCHAR (100)`	The name of the cache group. Specify `NULL` when installing objects for ASYNCHRONOUS WRITETHROUGH cache groups or to uninstall all Oracle objects in the AUTOREFRESH user's account.
install_flag	`TT_INTEGER NOT NULL`	If *install_flag* is 1, `ttCacheSqlGet` returns Oracle SQL to install the autorefresh or asynchronous writethrough Oracle objects. If *install_flag* is 0, `ttCacheSqlGet` returns SQL to uninstall the previously created objects.

Result set

`ttCacheSqlGet` returns the result set:

Column	Type	Description
retval	`TT_VARCHAR (4096) NOT NULL`	The Oracle SQL statement to uninstall or install autorefresh or asynchronous writethrough Oracle objects.

Column	Type	Description
continueFlag	TT_SMALLINT NOT NULL	Non zero only if the Oracle SQL statement in the *retval* result column exceeds 4096 bytes and must be continued into the next result row.

Example

```
CALL ttCacheSqlGet('INCREMENTAL_AUTOREFRESH', 'westernCustomers', 1);
```

To remove all Oracle objects in the AUTOREFRESH user's account, use:

```
CALL ttCacheSqlGet('INCREMENTAL_AUTOREFRESH', NULL, 0);
```

Notes

This procedure is available only for IMDB Cache.

Each returned *retval* field contains a separate Oracle SQL statement that may be directly executed on Oracle. A row may end in the middle of a statement, as indicated by the *continueFlag* field. In this case, the statement must be concatenated with the previous row to produce a usable SQL statement.

The script output of this procedure is not compatible with Oracle's SQL*Plus utility. However, you can use the `ttIsql` `cachesqlget` command to generate a script that is compatible with the SQL*Plus utility.

You can specify `NULL` for the *cache_group_name* option to generate Oracle SQL to clean up Oracle objects after a database has been destroyed by the `ttDestroy` utility.

ttCacheStart

Description

This procedure starts the TimesTen cache agent for the connected database.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheStart()
```

Parameters

`ttCacheStart` has no parameters.

Result set

`ttCacheStart` returns no results.

Examples

To start the TimesTen cache agent, use:

```
CALL ttCacheStart ();
```

Notes

This procedure is available only for IMDB Cache.

The cache administration user ID and password must be set before starting the cache agent when there are or might be `AUTOREFRESH` or `ASYNCHRONOUS WRITETHROUGH` cache groups in the database.

If you attempt to start the TimesTen cache agent (by changing the policy from manual to always) for a database with a relative path, TimesTen looks for the database relative to where the TimesTen Data Manager is running, and fails. For example, on Windows, if you specify the path for the database as `DataStore=./payroll` and attempt to start the TimesTen cache agent with this built-in procedure, the agent is not started because TimesTen Data Manager looks for the database in the `\srv` directory. On UNIX, the TimesTen Data Manager looks in the `/var/TimesTen/`*`instance`* directory.

When using this procedure, no application, including the application making the call, can be holding a connection that specifies database-level locking (`LockLevel=1`).

See also

`ttCacheConfig`
`ttCacheDbCgStatus`
`ttCachePolicyGet`
`ttCachePolicySet`
`ttCacheStop`
`ttCacheUidPwdSet`
`ttCacheUidGet`
`ttAdmin`

ttCacheStop

Description

This procedure stops the TimesTen cache agent for the connected database.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheStop(timeout)
```

Parameters

`ttCacheStop` has the parameter:

Parameter	Type	Description
timeout	`TT_INTEGER`	Specifies that the TimesTen daemon should kill the cache agent if it doesn't stop within *timeout* seconds. If set to 0, the daemon waits forever for the cache agent. The default value is 100 seconds.

Result set

`ttCacheStop` returns no results.

Examples

To stop the TimesTen cache agent, use:

```
CALL ttCacheStop();
```

Notes

This procedure is available only for IMDB Cache.

Do not shut down the cache agent immediately after dropping or altering a cache group. Instead, wait for at least two minutes. Otherwise, the cache agent may not get a chance to clean up the Oracle objects that were used by the AUTOREFRESH feature.

When using this procedure, no application, including the application making the call, can be holding a connection that specifies database-level locking (`LockLevel=1`).

See also

`ttCachePolicySet`
`ttCacheStart`
`ttCacheUidPwdSet`
`ttCacheUidGet`
`ttAdmin`

ttCacheUidGet

Description

This procedure returns the cache administration user ID for the database. If the cache administration user ID and password have not been set for the database, `ttCacheUidGet` returns `NULL`.

Required privilege

This procedure requires `CACHE_MANAGER` privilege.

Syntax

```
ttCacheUidGet()
```

Parameters

`ttCacheUidGet` has no parameters.

Result set

`ttCacheUidGet` returns the results:

Column	Type	Description
UID	`TT_VARCHAR (30)`	The current cache administration user ID, used for AUTOREFRESH and ASYNCHRONOUS WRITETHROUGH cache groups.

Example

```
CALL ttCacheUidGet();
```

Note

This procedure is available only for IMDB Cache.

See also

`ttCacheUidPwdSet`
`ttAdmin`

ttCacheUidPwdSet

Description

This procedure sets the cache administration user ID and password. You only need to specify the cache administration user ID and password once for each new database. The cache administration password can be changed at any time.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheUidPwdSet('UID', 'PWD')
```

Parameters

`ttCacheUidPwdSet` has these parameters:

Parameter	Type	Description
UID	TT_VARCHAR (30)	The cache administration user ID, used for AUTOREFRESH and ASYNCHRONOUS WRITETHROUGH cache groups.
PWD	TT_VARCHAR (30)	The password for the cache administration user.

Result set

`ttCacheUidPwdSet` returns no results.

Example

```
CALL ttCacheUidPwdSet('myid', 'mypwd');
```

Notes

This procedure cannot be called from a Client/Server connection.

This procedure is available only for IMDB Cache.

The cache administration user ID and password cannot be set while either the cache agent or the replication agent are running.

The cache administration user ID cannot be reset while there are ASYNCHRONOUS WRITETHROUGH cache groups or AUTOREFRESH cache groups (with a state that is not equal to `OFF`) on the database.

See also

`ttCacheUidGet`
`ttAdmin`

ttCkpt

Description

This procedure performs a non-blocking checkpoint operation. The blocking checkpoints are described in "ttCkptBlocking" on page 2-42. A checkpoint operation is used to make a record of the current state of the database on disk and to purge transaction log files. A non-blocking checkpoint does not require any locks on the database.

Applications should checkpoint databases periodically either by setting the background checkpointing attributes (`CkptFrequency` and `CkptLogVolume`) or by explicitly calling this procedure. Applications can call this procedure asynchronously to any other application running on the database.

By default, TimesTen performs background checkpoints at regular intervals.

In the case that your application attempts to perform a checkpoint operation while a backup is in process, the backup waits until the checkpoint finishes. Regardless of whether the checkpoint is a background checkpoint or an application-requested checkpoint, the behavior is:

- If a backup or checkpoint is running and you try to do a backup, it waits for the running backup or checkpoint to finish.
- If a backup or checkpoint is running and you try to do a checkpoint, it does not wait. It returns an error immediately.

To turn off background checkpointing, set `CkptFrequency=0` and `CkptLogVolume=0`.

When a database crashes, and the checkpoints on disk are non-blocking checkpoints, TimesTen uses the log to recover.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttCkpt()
```

Parameters

`ttCkpt` has these optional parameters:

Parameter	Type	Description
timeout	`TT_INTEGER`	The time (in seconds) that `ttCkpt` should wait to get a database lock before timing out. The value of *timeout* can be between 0 and one million, inclusively. If not specified, it defaults to infinity (the checkpoint never times out).
retries	`TT_INTEGER`	The number of times that `ttCkpt` should attempt to get a database lock, if timeouts occur. The value of *retries* can be between 0 and 10, inclusive. If not specified, defaults to zero.

Result set

`ttCkpt` returns no results.

Example

```
CALL ttCkpt();
```

Note

For a description of checkpoints, see "Transaction Management and Recovery" in *Oracle TimesTen In-Memory Database Operations Guide*.

See also

`ttCkptBlocking`
`ttCkptConfig`
`ttCkptHistory`

ttCkptBlocking

Description

This procedure performs a blocking checkpoint operation. A checkpoint operation is used to make a record of the current state of the database on disk, and to purge transaction log files. This checkpoint requires exclusive access to the database, and so may cause other applications to be blocked from the database while the checkpoint is in progress.

When this procedure is called, TimesTen performs a blocking checkpoint when the current transaction is committed or rolled back. If, at that time, other transactions are in progress, the checkpointing connection waits until the other transactions have committed or rolled back. While the checkpoint connection is waiting, any other new transactions that want to start form a queue behind the checkpointing transaction. As a result, if any transaction is long-running, it may cause many other transactions to be held up. So this blocking checkpoint should be used with caution. To perform a non-blocking checkpoint, use the `ttCkpt` procedure.

No log is needed to recover when blocking checkpoints are used. TimesTen uses the log, if present, to bring the database up to date after recovery.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttCkptBlocking(timeout, retries)
```

Parameters

`ttCkptBlocking` has these optional parameters:

Parameter	Type	Description
timeout	`TT_INTEGER`	The time (in seconds) that `ttCkptBlocking` should wait to get a database lock before timing out. The value of *timeout* can be between 0 and one million, inclusively. If not specified, it defaults to infinity (the checkpoint never times out).
retries	`TT_INTEGER`	The number of times that `ttCkptBlocking` should attempt to get a database lock, if timeouts occur. The value of *retries* can be between 0 and 10, inclusive. If not specified, defaults to zero.

Result set

`ttCkptBlocking` returns no results.

Example

```
CALL ttCkptBlocking();
CALL ttCkptBlocking(1,10);
```

Notes

Because the checkpoint takes place at commit or rollback, the call to `ttCkptBlocking` always succeed. At commit or rollback, any problems with the checkpoint operation, such as a lack of disk space or a timeout, result in a warning being returned to the application. Checkpoint problems are not reflected as errors, since the commit or rollback of which they are a part can succeed even if the checkpoint fails. Warnings are reflected in ODBC with the return code `SQL_SUCCESS_WITH_INFO`.

For more information on checkpoints, see "Transaction Management and Recovery" in *Oracle TimesTen In-Memory Database Operations Guide*.

See also

`ttCkpt`
`ttCkptConfig`
`ttCkptHistory`

ttCkptConfig

Description

This procedure reconfigures the background checkpointer dynamically or returns the currently active settings of the configuration parameters. Changes made using `ttCkptConfig` become effective immediately. Thus, changes to *`ckptRate`* can take effect on a checkpoint that is currently in progress.

Changes made to the background checkpointer using `ttCkptConfig` are persistent. Subsequent loads of the database retain the new settings, unless the `CkptFrequency` and `CkptLogVolume` connection attributes are specified in the DSN or connection string, in which case the attribute values are used instead.

Required privilege

This procedure requires no privilege to query the current values. It requires the `ADMIN` privilege to change the current values.

Syntax

```
ttCkptConfig(ckptFrequency, ckptLogVolume, ckptRate)
```

Parameters

`ttCkptConfig` has these parameters:

Parameter	Type	Description
`ckptFrequency`	`TT_INTEGER`	Checkpoint frequency in seconds. Values from 0 to MAXINT are allowed. A value of 0 means that checkpoint frequency is not considered when scheduling checkpoints.
`ckptLogVolume`	`TT_INTEGER`	Log volume between checkpoints in megabytes. Values from 0 to MAXINT are allowed. A value of 0 means that checkpoint log volume is not considered when scheduling checkpoints.
`ckptRate`	`TT_INTEGER`	Specifies the rate in MB per second at which a checkpoint should be written to disk. A value of 0 indicates that the rate should not be limited, a value of `NULL` means that the rate should be left unchanged. Changes to this parameter take effect even on a checkpoint that is currently in-progress.

Result set

`ttCkptConfig` returns these results:

Column	Type	Description
ckptFrequency	TT_INTEGER NOT NULL	Currently active setting for checkpoint frequency in seconds.
ckptLogVolume	TT_INTEGER NOT NULL	Currently active setting for log volume between checkpoints in Megabytes.
ckptRate	TT_INTEGER NOT NULL	Current rate at which checkpoints are written to disk.

Examples

To view the current settings of the background checkpointer configuration parameters, use:

```
CALL ttCkptConfig;
< 600, 32, 0 >
1 row found.
```

To stop the background checkpointer from initiating checkpoints unless the log reaches its limit, use:

```
CALL ttCkptConfig(0);
< 0, 32, 0 >
1 row found.
```

To stop the background checkpointer from initiating checkpoints, use:

```
CALL ttCkptConfig(NULL, 0);
< 0, 0, 0 >
1 row found.
```

To set the background checkpointer configuration to initiate a checkpoint every 600 seconds or to checkpoint when the log reaches 32 megabytes (whichever comes first), use:

```
CALL ttCkptConfig(600, 32);
< 600, 32, 0 >
1 row found.
```

Notes

By default, TimesTen performs background checkpoints at regular intervals.

In the case that your application attempts to perform a checkpoint operation while a backup is in process, the backup waits until the checkpoint finishes. Regardless of whether the checkpoint is a background checkpoint or an application-requested checkpoint, the behavior is:

- If a backup or checkpoint is running and you try to do a backup, it waits for the running backup or checkpoint to finish.
- If a backup or checkpoint is running and you try to do a checkpoint, it does not wait. It returns an error immediately.

To turn off background checkpointing, set `CkptFrequency=0` and `CkptLogVolume=0`.

See also

CkptFrequency
CkptLogVolume
ttCkpt
ttCkptHistory

ttCkptHistory

Description

This procedure returns information about the last eight checkpoints of any type taken by any agent.

Required privilege

This procedure requires no privilege.

Syntax

```
ttCkptHistory( )
```

Parameters

`ttCkptHistory` has no parameters.

Result set

`ttCkptHistory` returns the result set:

Column	Type	Description
startTime	TT_TIMESTAMP NOT NULL	Time when the checkpoint was begun.
endTime	TT_TIMESTAMP	Time when the checkpoint completed.
type	TT_CHAR (16) NOT NULL	The type of checkpoint taken. Value is one of: Static - Automatically taken at database creation and at last disconnect. Blocking - Transaction-consistent checkpoint. Fuzzy - Non-blocking checkpoint. The background checkpointer performs this type if possible. None - For `ReplicationParallelism` databases, which have no checkpoint files.

Column	Type	Description
status	TT_CHAR (16) NOT NULL	Result status of the checkpoint operation. Value is one of: In Progress - The checkpoint is currently in progress. Only the most recent result row can have this status. Completed - The checkpoint completed successfully. Failed - The checkpoint failed. Only the most recent result row can have this status. In this case the error column indicates the reason for the failure.
initiator	TT_CHAR (16) NOT NULL	The source of the checkpoint request. Value is one of: User - A user-level application. This includes TimesTen utilities such as `ttIsql`. Checkpointer - The background checkpointer. Subdaemon - The managing subdaemon of the database. For a shared database, the final disconnect checkpoint is taken by the subdaemon.
error	TT_INTEGER	If a checkpoint fails, this column indicates the reason for the failure. The value is one of the TimesTen error numbers.
ckptFileNum	TT_INTEGER NOT NULL	The database file number used by the checkpoint. This corresponds to the number in the checkpoint file extension *datastore*.ds0 or *datastore*.ds1.
ckptLFN	TT_INTEGER	The transaction log file number of the checkpoint log record.
ckptLFO	TT_INTEGER	The transaction log file offset of the checkpoint log record.
blksTotal	TT_BIGINT	The number of permanent blocks currently allocated in the database. These blocks are subject to consideration for checkpointing.
bytesTotal	TT_BIGINT	The number of bytes occupied by *blksTotal*.
blksInUse	TT_BIGINT	Of *blksTotal*, the number of blocks currently in use.
bytesInUse	TT_BIGINT	The number of bytes occupied by *blksInUse*.

Column	Type	Description
blksDirty	TT_BIGINT	The number of dirty blocks written by this checkpoint.
bytesDirty	TT_BIGINT	The number of bytes occupied by *blksDirty*.
bytesWritten	TT_BIGINT	The total number of bytes written by this checkpoint.
Percent_Complete	TT_INTEGER	If there is an in-progress checkpoint, indicates the percentage of the checkpoint that has been completed. If no checkpoint is in-progress, the value is NULL. The returned value is calculated by comparing the block ID of the last-written block against the database's PermSize. The value does not necessarily indicate the precise time remaining to complete the checkpoint, although it does give some indication of the remaining time needed to complete the disk write. The field shows only the progress of the writing of dirty blocks and does not include additional bookkeeping at the end of the checkpoint.

Examples

This example shows a checkpoint in progress:

```
Call ttckpthistory;
< 2010-04-14 16:56:34.169520, <NULL>, Fuzzy         , In Progress    , User
, <NULL>, 0, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>,
<NULL>, 13, 6 >

< 2010-04-14 16:55:47.703199, 2010-04-14 16:55:48.188764, Fuzzy           ,
Completed       , Checkpointer    , <NULL>, 1, 0, 8964304, 294, 33554432, 291,
5677288, 27, 1019512, 1065408, <NULL>, 5 >

< 2010-04-14 16:54:47.106110, 2010-04-14 16:54:47.723379, Static          ,
Completed       , Subdaemon       , <NULL>, 0, 0, 8960328, 294, 33554432, 291,
5677288, 256, 33157172, 5321548, <NULL>, 4 >

< 2010-04-14 16:54:41.633792, 2010-04-14 16:54:42.568469, Blocking        ,
Completed       , User            , <NULL>, 1, 0, 8958160, 294, 33554432, 291,
5677288, 31, 1162112, 6604976, <NULL>, 3 >

< 2010-04-14 16:54:37.438827, 2010-04-14 16:54:37.977301, Static          ,
Completed       , User            , <NULL>, 0, 0, 1611984, 93, 33554432, 92,
1853848, 93, 33554432, 1854052, <NULL>, 2 >

< 2010-04-14 16:54:36.861728, 2010-04-14 16:54:37.438376, Static          ,
Completed       , User            , <NULL>, 1, 0, 1609936, 93, 33554432, 92,
1853848, 93, 33554432, 1854052, <NULL>, 1 >

6 rows found.
```

This example shows that an error occurred during the most recent checkpoint attempt:

```
call ttckpthistory;
< 2010-04-14 16:57:14.476860, 2010-04-14 16:57:14.477957, Fuzzy       , Failed
, User            , 847, 1, <NULL>, <NULL>, 0, 0, 0, 0, 0, 0, 0, <NULL>, 7 >

< 2010-04-14 16:56:34.169520, 2010-04-14 16:56:59.715451, Fuzzy           ,
Completed       , User            , <NULL>, 0, 0, 8966472, 294, 33554432, 291,
5677288, 5, 522000, 532928, <NULL>, 6 >

< 2010-04-14 16:55:47.703199, 2010-04-14 16:55:48.188764, Fuzzy           ,
Completed       , Checkpointer    , <NULL>, 1, 0, 8964304, 294, 33554432, 291,
5677288, 27, 1019512, 1065408, <NULL>, 5 >

< 2010-04-14 16:54:47.106110, 2010-04-14 16:54:47.723379, Static          ,
Completed       , Subdaemon       , <NULL>, 0, 0, 8960328, 294, 33554432, 291,
5677288, 256, 33157172, 5321548, <NULL>, 4 >

< 2010-04-14 16:54:41.633792, 2010-04-14 16:54:42.568469, Blocking        ,
Completed       , User            , <NULL>, 1, 0, 8958160, 294, 33554432, 291,
5677288, 31, 1162112, 6604976, <NULL>, 3 >

< 2010-04-14 16:54:37.438827, 2010-04-14 16:54:37.977301, Static          ,
Completed       , User            , <NULL>, 0, 0, 1611984, 93, 33554432, 92,
1853848, 93, 33554432, 1854052, <NULL>, 2 >

< 2010-04-14 16:54:36.861728, 2010-04-14 16:54:37.438376, Static          ,
Completed       , User            , <NULL>, 1, 0, 1609936, 93, 33554432, 92,
1853848, 93, 33554432, 1854052, <NULL>, 1 >

7 rows found.
```

Notes

Results are ordered by start time, with the most recent first.

A failed row is overwritten by the next checkpoint attempt.

See also

ttCkpt
ttCkptBlocking

ttCompact

Description

This procedure compacts both the permanent and temporary data partitions of the database.

ttCompact merges adjacent blocks of free space, but does not move any items that are allocated. Therefore, fragmentation that is caused by small unallocated blocks of memory surrounded by allocated blocks of memory is not eliminated by using `ttCompact`.'

This procedure is supported for backward compatibility. New applications should not call it.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttCompact()
```

Parameters

`ttCompact` has no parameters.

Result set

`ttCompact` returns no results.

Example

```
CALL ttCompact;
```

Note

Compacting data does not modify result addresses.

See also

`ttCompactTS`

ttCompactTS

Description

This procedure is similar to ttCompact, except that ttCompactTS may be used to compact a small fraction of the database, while ttCompact compacts the entire database. ttCompactTS is a time-sliced version of ttCompact. ttCompactTS iterates through all the blocks in the database compacting the quantum specified each time. When a sweep is completed, the value of the DS_COMPACTS field in the MONITOR table is incremented.

This procedure is supported for backward compatibility. New applications should not call it.

Required privilege

This procedure requires the ADMIN privilege.

Syntax

```
ttCompactTS(quantum)
```

Parameters

ttCompactTS has the parameter:

Parameter	Type	Description
quantum	TT_INTEGER NOT NULL	A nonzero positive integer that specifies the number of data blocks a ttCompactTS should compact. Each quantum corresponds to one data block.

Result set

ttCompactTS returns no results.

Example

```
CALL ttCompactTS (5);
```

Note

Compacting data does not modify result addresses.

See also

ttCompact

ttConfiguration

Description

This procedure returns the values for most attributes for the current database connection.

Required privilege

This procedure requires no privilege.

Syntax

```
ttConfiguration('paramName')
```

Parameters

`ttConfiguration` has the optional parameter:

Parameter	Type	Description
paramName	`TT_VARCHAR (30)`	The name of a connection attribute for which you want this procedure to return the value.

Result set

`ttConfiguration` returns the result set:

Column	Type	Description
paramName	`TT_VARCHAR (30) NOT NULL`	The names of the connection attributes specified in the connection string, returned in alphabetical order.
paramValue	`TT_VARCHAR (1024)`	The values of the connection attributes specified in the connection string.

Example

To see the value of the `QueryThreshold` attribute, use

```
CALL ttConfiguration('querythreshold');
<QueryThreshold, 0>
1 row found
```

To see the values of all attributes, use:

```
CALL ttConfiguration();
< CacheGridEnable, 1 >
< CacheGridMsgWait, 60 >
< CkptFrequency, 600 >
< CkptLogVolume, 0 >
. . .
```

Note

Client driver attributes are not returned by this procedure.

See also

"Connection Attributes" on page 1-1

ttContext

Description

This procedure returns the context value of the current connection as a BINARY(8) value. The context can be used to correlate a unique connection to a database from the list of connections presented by the `ttStatus` utility and the `ttDataStoreStatus` built-in procedure.

Required privilege

This procedure requires no privilege.

Syntax

```
ttContext()
```

Parameters

`ttContext` has no parameters.

Result set

`ttContext` returns the result set:

Column	Type	Description
CONTEXT	BINARY(8)	Current connection's context value.

Example

```
CALL ttContext;
```

Note

The context value numbers are unique only within a process. The context value number is not unique within the entire database. Therefore you may see the same context value number for different processes.

See also

`ttDataStoreStatus`
`ttStatus`

ttDataStoreStatus

Description

This procedure returns the list of processes connected to a database. If the *dataStore* parameter is specified as NULL, then the status of all active databases is returned.

The result set is similar to the printed output of the ttStatus utility.

Required privilege

This procedure requires no privilege.

Syntax

```
ttDataStoreStatus('dataStore')
```

Parameters

ttDataStoreStatus has the parameter:

Parameter	Type	Description
dataStore	TT_VARCHAR (256)	Full path name of desired database or NULL for all databases.

Result set

ttDataStoreStatus returns the result set:

Column	Type	Description
dataStore	TT_VARCHAR (256) NOT NULL	Full path name of database.
PID	TT_INTEGER NOT NULL	Process ID.
Context	BINARY(8) NOT NULL	Context value of connection.
conType	TT_CHAR (16) NOT NULL	Type of process connected. The result can be one of the following: **application** - an ordinary application is connected. **replication** - a replication agent is connected. **subdaemon** - a subdaemon is connected. **oracleagent** - an cache agent is connected.
ShmID	TT_VARCHAR (260) NOT NULL	A printable version of the shared memory ID that the database occupies.
connection_Name	TT_CHAR (30) NOT NULL	The symbolic name of the database connection.
connID	TT_INTEGER NOT NULL	The numeric ID of the database connection

Example

```
CALL ttDataStoreStatus('/data/Purchasing');
```

See also

ttContext
ttStatus

ttDurableCommit

Description

This procedure specifies that the current transaction should be made durable when it is committed. It only has an effect if the application is connected to the database with `DurableCommits` disabled and logging to disk enabled. (See "Logging" on page 1-35.)

Calling `ttDurableCommit` also makes the current transaction and any previously committed non-durable transactions durable. There is no effect on other transactions that are committed subsequent to calling `ttDurableCommit`. `ttDurableCommit` does not commit transactions. The application must do the commit, for example with a call to `SQLTransact`.

Required privilege

This procedure requires no privilege.

Syntax

```
ttDurableCommit()
```

Parameters

`ttDurableCommit` has no parameters.

Result set

`ttDurableCommit` returns no results.

Example

```
CALL ttDurableCommit;
```

Note

Some controllers or drivers may only write data into cache memory in the controller or may write to disk some time after the operating system is told that the write is done. In these cases, a power failure may mean that some information you thought was durably committed does not survive the power failure. To avoid this loss of data, configure your disk to write all the way to the recording media before reporting completion or you can use an Uninterruptable Power Supply.

ttGridAttach

Description

This procedure attaches a grid member to an existing local cache grid. A grid member can be a standalone TimesTen database or a TimesTen active standby pair.

If a member is an active standby pair, both nodes of the pair must attach to the grid. When calling the `ttGridAttach` built-in procedure from each node of the active standby pair, specify the IP address or host name of both nodes.

This procedure starts the cache agent if it is not already running. This procedure cannot be used remotely.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

For a standalone TimesTen database:

```
ttGridAttach(currentNode, 'name1', IPAddr1, port1)
```

For a node of an active standby pair:

```
ttGridAttach(currentNode, 'name1', IPAddr1, port1 'name2', IPAddr2, port2)
```

Parameters

`ttGridAttach` has the parameters:

Parameter	Type	Description
currentNode	TT_INTEGER NOT NULL	The node number for the master database. Valid values for this parameter are: **1** - Standalone or active master database **2** - Standby master database.
name1	TT_VARCAHR (30)	Fully qualified name that uniquely identifies the grid member for the active master database.
IPAddr1	`TT_VARCHAR (128) NOT NULL`	IP address of the node where the active master database resides.
port1	TT_INTEGER NOT NULL	Port number for the cache agent process of the active master database or a standalone database.
name2	TT_VARCAHR (30)	Fully qualified name that uniquely identifies the grid member for the standby master database.
IPAddr2	`TT_VARCHAR (128) NOT NULL`	IP address of the node where the standby master database resides.
port2	TT_INTEGER NOT NOLL	Port number for the cache agent process of the standby master database.

Result set

`ttGridAttach` returns no results.

Examples

To attach to a standalone TimesTen database to a grid:

```
CALL ttGridAttach (1, 'alone2','sys2',5002);
```

To attach an active master database to a grid:

```
CALL ttGridAttach(1,'cacheact','sys1',5003,'cachestand','sys2',5004);
```

To attach a standby master database to a grid:

```
CALL ttGridAttach(2,'cacheact','sys1',5003,'cachestand','sys2',5004);
```

Note that the only difference between the calls for attaching the active and the standby master stores is the node number.

See also

`ttGridCheckOwner`
`ttGridCreate`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachList`
`ttGridDetachAll`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridCheckOwner

Description

This procedure checks if the number of rows in global cache groups match number of rows in the ownership tables. Call this procedure only when the cache grid is quiet.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridCheckOwner(['cvName', 'cvOwner'])
```

Parameters

`ttGridCheckOwner` has the optional parameter:

Parameter	Type	Description
cvName	TT_VARCHAR (30)	The name of the cache group to be checked...name and owner of cache group that must be checked. If null, all cache groups are checked.
cvOwner	TT_VARCHAR (30)	.The owner of the cache group to be checked. If null, all cache groups are checked.

Result set

`ttGridCheckOwner` displays no results.

Example

To get information on the `mygroup` cache group, owned by user terry, use:

```
CALL ttGridCheckOwner ('mygroup', 'terry');
```

To get information on all cache groups, use:

```
CALL ttGridCheckOwner();
```

See also

`ttGridAttach`
`ttGridCreate`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridCreate

Description

This procedure creates a cache grid. This built-in procedure must be run only once. You can run it from any standalone database or from the active or standby master database in an active standby pair.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridCreate('gridName')
```

Parameters

`ttGridCreate` has the parameter:

Parameter	Type	Description
gridName	`TT_VARCHAR (30) NOT NULL`	Specifies the name of the grid

Result set

`ttGridCreate` returns no results.

Example

To create a grid named `mygrid`:

```
CALL ttGridCreate ('mygrid');
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridDestroy

Description

This procedure destroys a cache grid by removing all cache grid objects stored on the Oracle database.

By default, this built-in procedure does not destroy the grid if there are still attached members or existing global cache groups. Before destroying a cache grid, detach all the TimesTen databases from the cache grid. To force the grid to be destroyed, supply a value of '1' as an argument to the `force` parameter.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridDestroy('gridName', [force])
```

Parameters

`ttGridDestroy` has the parameters:

Parameter	Type	Description
gridName	`TT_VARCHAR (30) NOT NULL`	The fully qualified name of the grid to be destroyed.
force	`TT_INTEGER`	This optional parameter forces the cache grid to be destroyed even if there are still grid members attached to the cache grid or if it still contains global cache groups. Valid value is 1.

Result set

`ttGridDestroy` returns no results.

Example

To destroy the `mygrid` cache grid with force, use:

```
CALL ttGridDestroy ('mygrid', 1);
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridCreate`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridDetach

Description

This procedure detaches a node from a cache grid.

Use this procedure before destroying a cache grid. You cannot destroy a cache grid if there are any nodes attached to the cache grid.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridDetach([['nodeMemberName',] [force]])
```

Parameters

`ttGridDetach` has the optional parameters:

Parameter	Type	Description
nodeMemberName	`TT_VARCHAR (200)`	Specifies the node to detached from the grid. Each node of an active standby pair must be detached separately.
force	`TT_INTEGER`	This optional parameter forces a node to be detached without checking whether it is terminated. Valid value is 1.

Result set

`ttGridDetach` returns no results.

Example

To detach the current node from the grid, use

```
CALL ttGridDetach();
```

To detach the remote node `TTGRID_alone2_2` from the grid, use

```
CALL ttGridDetach('TTGRID_alone2_2',1);
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridCreate`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridDestroy`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridDetachAll

Description

This procedure detaches all attached members from the grid. A grid member can be a standalone TimesTen database or a TimesTen active standby pair.

This procedure starts the cache agent if it is not already running.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridDetachAll()
```

Parameters

`ttGridDetachAll` has no parameters:

Result set

`ttGridDetachAll` returns no results.

Examples

To detach all grid members, use:

```
CALL ttGridDetachAll();
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridCreate`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachList`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridDetachList

Description

This procedure detaches the nodes in the list. It is useful for remote nodes, because they are unavailable.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridDetachList('nodeMemberName1 [nodeMemberName2 ...]' [,force])
```

Parameters

`ttGridDetachList` has the parameters:

Parameter	Type	Description
nodeMemberName	`TT_VARCHAR (8192) NOT NULL`	The fully qualified name of the node to be removed
force	`TT_INTEGER`	This optional parameter forces nodes to be detached without checking whether they are terminated. Valid value is 1.

Result set

`ttGridDetachList` returns no results.

Example

```
CALL ttGridDetachList('TTGRID_cacheact_3A TTGRID_cachestand_3B',1);
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridCreate`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDestroy`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridGlobalCGResume

Description

This procedure resumes operations that were blocked after a call to `ttGridGlobalCGSuspend`. This procedure starts the cache agent if it is not already running.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridGlobalCGResume()
```

Parameters

`ttGridGlobalCGResume` has no parameters.

Result set

`ttGridGlobalCGResume` returns no results.

Examples

To detach all grid members, use:

```
CALL ttGridGlobalCGResume();
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridCreate`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridGlobalCGSuspend

Description

This procedure temporarily blocks dynamic loading and deleting cache instances for global cache groups. This procedure starts the cache agent if it is not already running. Use the `ttGridGlobalCGResume` procedure to re-enable these actions.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridGlobalCGSuspend(wait)
```

Parameters

`ttGridGlobalCGSuspend` has the parameter:

Parameter	Type	Description
wait	`TT_INTEGER NOT NULL`	The number of seconds that the command waits for a pending delete to be propagated to Oracle or a pending transparent load operation to complete before returning. TimesTen returns an error if either the pending delete or the pending transparent load operation cannot complete in the specified time. If no value is specified, there is no wait interval.

Result set

`ttGridGlobalCGSuspend` returns no results.

Examples

To set a wait interval of 10 seconds:

```
CALL ttGridGlobalCGSuspend(10);
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridCreate`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridGlobalCGResume`
`ttGridInfo`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridInfo

Description

This procedure returns information about the specified cache grid or all cache grids.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridInfo(['gridName'])
```

Parameters

`ttGridInfo` has the optional parameter:

Parameter	Type	Description
gridName	`TT_VARCHAR (30)`	If *gridName* is specified, displays information about the specified grid. Otherwise, displays information about all grids.

Result set

`ttGridInfo` returns information about the cache grid.

Column	Type	Description
gridName	`TT_VARCHAR (30)`	The name of the grid specified
cacheAdminID	`TT_VARCHAR (30)` `NOT NULL`	The cache administration user ID associated with the grid.
platform	`TT_VARCHAR (100)`	The operating system platform on which the grid is operating. The platform value is displayed as: *operating system*,*bit-level* For example: `<. . ., Solaris x86, 64-bit, . . .>` The comma does not represent a new column.
major1, major2, major3	`TT_VARCHAR (10)` for each field	The major TimesTen release associated with the grid. For example, release 11.2.1 is represented as 11, 2, 1.

Example

To get information on the `mygrid` cache grid, use:

```
CALL ttGridInfo ('mygrid');
< MYGRID, CACHEUSER, Linux Intel x86, 32-bit, 11, 2, 1 >
```

To get information on all grids, use:

```
CALL ttGridInfo();
```

See also

ttGridAttach
ttGridCheckOwner
ttGridCreate
ttGridDestroy
ttGridDetach
ttGridDetachAll
ttGridDetachList
ttGridGlobalCGResume
ttGridGlobalCGSuspend
ttGridNameSet
ttGridNodeStatus
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridNameSet

Description

This procedure associates a TimesTen database with a grid.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridNameSet('gridName')
```

Parameters

`ttGridNameSet` has the parameter:

Parameter	Type	Description
gridName	`TT_VARCHAR (30)`	Associates the TimesTen database that calls the procedure with the grid specified by *gridName*.

Result set

`ttGridNameSet` returns no results.

Example

To associate the database with the grid `mygrid`, use.

```
CALL ttGridNameSet('mygrid');
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridCreate`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttGridNodeStatus

Description

This procedure returns information about all members of the specified cache grid. If no grid name is specified, then it displays information about all members of all cache grids associated with the Oracle database.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttGridNodeStatus(['gridName'])
```

Parameters

`ttGridNodeStatus` has the optional parameter:

Parameter	Type	Description
gridName	`TT_VARCHAR (30)`	If *gridName* is specified, displays information about all members of the named grid. Otherwise, displays information about all grids.

Result Set

`ttGridNodeStatus` returns the results:

Column	Type	Description
gridName	`TT_VARCHAR (30)`	The name of the grid.
memberID	`TT_INTEGER NOT NULL`	The unique ID of the grid member.
activeNode	`TT_INTEGER NOT NULL`	The number of the node on which the active master database or a standalone database currently resides.
node1Attached	`CHAR (1) NOT NULL`	Indicates if the active node is attached to the grid: T - the active is attached. F - the active is detached.
Host1	`TT_VARCHAR (200) NOT NULL`	The host name where the active database is located.
nodeName1	`TT_VARCHAR (200) NOT NULL`	The unique node name for the stand alone database or active standby database.
IPaddr1	`TT_VARCHAR (128) NOT NULL`	The IP address where the active master or stand alone database is located.
port1	`TT_INTEGER NOT NULL`	The port number for the cache agent process of the active master or stand alone database.

Column	Type	Description
node2Attached	CHAR (1)	Indicates if the standby node is attached to the grid: T - the standby is attached. F - the standby is detached.
host2	TT_VARCHAR (200)	The host name where the standby master database is located.
nodeName2	TT_VARCHAR (200)	The unique node name for the stand alone database or active standby database.
IPaddr2	TT_VARCHAR (128)	The IP address where the standby master database is located.
port2	TT_INTEGER	The port number for the cache agent process of the standby master database.

For a grid member that is a standalone database, the number of columns in the result set is fewer than for a member that is an active standby pair.

Example

If `ttgrid` is the only cache grid in the database, display information about its members:

```
Command> call ttGridNodeStatus;

< TTGRID, 1, 1, T, sys1, TTGRID_alone1_1, 140.87.0.201, 5001, <NULL>,
<NULL>,<NULL>, <NULL>, <NULL> >
< TTGRID, 2, 1, T, sys2, TTGRID_alone2_2, 140.87.0.202, 5002, <NULL>,
<NULL>,<NULL>, <NULL>, <NULL> >
< TTGRID, 3, 1, T, sys3, TTGRID_cacheact_3A, 140.87.0.203, 5003, T, sys4,
TTGRID_cachestand_3B, 140.87.0.204, 5004 >
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridCreate`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridInfo`
`ttGridNameSet`
"Configuring a cache grid" in *Oracle In-Memory Database Cache User's Guide*

ttHostNameGet

Description

This procedure returns the name of the current local host for the database. The value returned is only for the current session. It is not a systemwide setting and does not persist after the current session has been disconnected.

This procedure can be used to check whether a particular store name in a scheme refers to the current host. This can be helpful when configuring replication schemes.

Required privilege

This procedure requires no privilege.

Syntax

```
ttHostnameGet()
```

Parameters

`ttHostNameGet` has no parameters.

Result set

`ttHostNameGet` returns the result:

Column	Type	Description
hostName	`TT_VARCHAR (200)`	The current default local host setting for the database. If a default has not been supplied then the current hostname is returned.

Example

```
CALL ttHostNameGet ();
```

See also

`ttHostNameSet`
"Setting Up a Replicated System" in *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*

ttHostNameSet

Description

This procedure specifies the name of the default local host for the current database. The value is only used in the current session, it is not a systemwide setting and does not persist after the current session has been disconnected.

To configure master/subscriber relationships and replication object permissions correctly, Replication DDL processing relies on being able to determine whether a host name used in a replication scheme refers to the computer on which the script is currently being run. This procedure allows an application to set a default host name for the current session that can be used by Replication DDL processing whenever there is a need to establish the name of the current host.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttHostnameSet('hostName')
```

Parameters

`ttHostNameSet` has the parameter:

Parameter	Type	Description
hostName	`TT_VARCHAR (200)`	The required default name for the local computer. If `NULL` is supplied the default value is cleared.

Result set

`ttHostNameSet` returns no results.

Example

```
CALL ttHostNameSet ('alias1');
```

Note

The legal value of hostName can be any host name or IP address string except "`localhost`", "`127.0.0.1`" or "`::1`". You cannot set the default host name to a value that is different from a local host name used in an existing replication scheme.

See also

`ttHostNameGet`
"Setting Up a Replicated System" in *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*

ttLockLevel

Description

Changes the lock level between row-level and database-level locking on the *next* transaction and for all subsequent transactions for this connection. Applications can change the lock level again by calling `ttLockLevel` again. The initial value depends on the `LockLevel` attribute. See `LockLevel` for full details of the different locking levels.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttLockLevel('lockLevel')
```

Parameters

`ttLockLevel` has the parameter:

Parameter	Type	Description
lockLevel	`TT_CHAR (20) NOT NULL`	Locking level for the connection.

The value of *lockLevel* may be one of two case-insensitive strings:

`Row`: Locking should be set to row-level locking.

`DS`: Locking should be set to database-level locking.

Result set

`ttLockLevel` returns no results.

Example

```
CALL ttLockLevel ('Row');
```

Notes

This procedure does not affect the current transaction.

Row-level locking is required when caching Oracle tables.

This procedure must be called from within a transaction. It has the effect of setting the locking level for subsequent transactions for the connection that invoked it. The new lock level does not affect the current transaction. It takes effect at the beginning of the next transaction.

See also

`ttLockWait`

ttLockWait

Description

This procedure allows an application to change the lock timeout interval of the current connection. The change takes effect immediately and applies to all subsequent statements in the current transaction and all subsequent transactions on the connection.

The lock wait interval is the number of seconds to wait for a lock when there is contention on it. You can also indicate a fraction of a second.

Lock wait intervals are imprecise, and may be exceeded, generally by no more than 100 milliseconds, due to the scheduling of the agent that detects timeouts. This imprecision does not apply to zero second timeouts, which are always reported immediately.

Cache grid uses message wait time with lock wait time. When using cache grid, lock wait times are approximately half the value you have specified. If your applications require the full lock wait time, specify twice the desired seconds.

Required privilege

This procedure requires no privilege.

Syntax

```
ttLockWait(seconds)
```

Parameters

`ttLockWait` has the required parameters:

Parameter	Type	Description
seconds	`NUMBER (8,1) NOT NULL`	Number of seconds to wait for a lock when there is contention on it. You can also specify fractions of a second. Values between 0.0 and 1000000.0 inclusive are accepted.

Result set

`ttLockWait` returns no results.

Examples

To indicate a six second lock wait, use:

```
CALL ttLockWait (6);
```

To indicate a tenth of a second lock wait, use:

```
CALL ttLockWait (0.1);
```

Notes

When a lock is not immediately available to a TimesTen transaction, it waits a predetermined amount of time to try to get the lock. After that it times out the lock request and returns TimesTen error 6003 to the application. By default, TimesTen uses a value of 10 seconds for lock timeouts.

Of special interest is the lock time-out interval value of 0. If that value is specified, transactions do not wait for any unavailable locks. If the lock is not available, the request returns with TimesTen error 6003.

See also

`ttLockLevel`
"LockLevel" on page 1-54

ttLogHolds

Description

This procedure returns information about transaction log holds, including those created on behalf of incremental backups, replication peers, persistent XLA subscribers, XA, long-running transactions and checkpoints. This procedure can help diagnose situations where it appears that checkpoint operations are not purging all unneeded transaction log files.

Required privilege

This procedure requires no privilege.

Syntax

```
ttLogHolds()
```

Parameters

`ttLogHolds` has no parameters.

Result set

`ttLogHolds` returns the result set:

Column	Type	Description
HoldLFN	`TT_INTEGER NOT NULL`	Returns the transaction log file number of the hold.
HoldLFO	`TT_INTEGER NOT NULL`	Returns the transaction log file offset of the hold
type	`TT_CHAR (30) NOT NULL`	Returns the type of hold, one of: ■ Checkpoint ■ Replication ■ Backup ■ XLA ■ Long-Running Transaction ■ Long-Running XA Transaction

Column	Type	Description
description	TT_VARCHAR (1024) NOT NULL	Describes the type-specific object for which the hold was created. Each description corresponds with the Type returned. Descriptions are one of: ■ The name of the checkpoint file ■ The name of the replication subscriber ■ The parallel replication track ID used by the subscriber ■ The backup path ■ The name of the persistent XLA subscription and the process ID of the last process to open it, if it is open ■ The XA XID (transaction ID) of the XA transaction ■ The TimesTen transaction ID of the long-running transaction

Example

```
CALL ttLogHolds();
< 0, 1148544, Long-Running XA Transaction , 0x1-476c6f62616c-5861637431 >
< 0, 1149752, Long-Running Transaction, 4.2 >
< 0, 1149992, Checkpoint , sample.ds1 >
< 0, 1150168, Checkpoint , sample.ds0 >
```

ttMonitorHighWaterReset

Description

This procedures set the value of PERM_IN_USE_HIGH_WATER column in the MONITOR table to the current value of PERM_IN_USE_SIZE attribute and sets the value of the TEMP_IN_USE_HIGH_WATER column in the MONITOR table to the current value of TEMP_IN_USE_SIZE attribute. These columns are useful for sizing databases during application development and deployment.

Required privilege

This procedure requires the ADMIN privilege.

Syntax

```
ttMonitorHighWaterReset()
```

Parameters

ttMonitorHighWaterReset has no parameters.

Result set

ttMonitorHighWaterReset returns no results.

Example

```
CALL ttMonitorHighWaterReset();
```

ttOptClearStats

Description

This procedure clears the statistics for the specified table, causing the TimesTen query optimizer to use estimates or default values for subsequent queries involving the table. The procedure is useful if statistics are assumed to be out of date and an application wants to use built-in default values. This procedure removes all rows from the TBL_STATS and COL_STATS system tables that pertain to the specified tables. See "SYS.TBL_STATS" and "SYS.COL_STATS" in *Oracle TimesTen In-Memory Database System Tables and Limits Reference*.

Required privilege

This procedure requires no privilege for the table owner. This procedure requires no privilege if `tblName` is not specified, because the procedure operates on the current user's tables if `tblName` is not specified.

This procedure requires the `ALTER ANY TABLE` privilege if user is not the table owner.

Syntax

```
ttOptClearStats('tblName', invalidate)
```

Parameters

`ttOptClearStats` has these parameters:

Parameter	Type	Description
tblName	`TT_CHAR (61)`	Name of an application table. Can include table owner. If *tblName* is the empty string or is not specified, statistics are cleared for all the current user's tables in the database. Using a synonym to specify a table name is not supported.
invalidate	`TT_INTEGER`	0 (no) or 1 (yes). Default is 0. If *invalidate* is 1, all commands that reference the affected tables are reprepared automatically when they are re-executed, including commands prepared by other users. If *invalidate* is 0, the statistics are not considered modified and existing commands are not reprepared.

Result set

`ttOptClearStats` returns no results.

Example

```
CALL ttOptClearStats ( 'SALLY.ACCTS', 1 );
```

Clears the statistics for the `SALLY.ACCTS` table and reprepares all commands that affect the `ACCTS` table.

```
CALL ttOptClearStats();
```

Clears the statistics for all the current user's tables and reprepares all commands that affect these tables.

```
CALL ttOptClearStats('', 0);
```

Clears the statistics for all the current user's tables without repreparing commands that reference these tables.

See also

`ttOptEstimateStats`
`ttOptSetColIntvlStats`
`ttOptSetFlag`
`ttOptSetOrder`
`ttOptSetTblStats`
`ttOptUpdateStats`
`ttPLSQLMemoryStats`

ttOptEstimateStats

Description

This procedure updates the statistics for the specified table. This procedure estimates statistics by looking at a random sample of the rows in the specified table(s). The sample size is the number of rows specified (if *sampleStr* has the form '*n* ROWS') or a percentage of the total number of rows (if *sampleStr* has the form '*p* PERCENT').

Required privilege

This procedure requires no privilege for the table owner. This procedure requires no privilege if *tblName* is not specified, because the procedure operates on the current user's tables if *tblName* is not specified.

This procedure requires the ALTER ANY TABLE privilege if user is not the table owner.

Syntax

```
ttOptEstimateStats('tblName', invalidate, 'sampleStr')
```

Parameters

ttOptEstimateStats has these parameters:

Parameter	Type	Description
tblName	TT_CHAR(61)	Name of an application table. Can include table owner. If *tblName* is the empty string, statistics are estimated for all the current user's tables in the database. Using a synonym to specify a table name is not supported.
invalidate	TT_INTEGER	0 (no) or 1 (yes). If *invalidate* is 1, all commands that reference the affected tables are automatically prepared again when re-executed, including commands prepared by other users. If *invalidate* is 0, the statistics are not considered to have been modified and existing commands are not reprepared. The *invalidate* parameter is optional and defaults to 0.
sampleStr	TT_VARCHAR (255) NOT NULL	String of the form '*n* ROWS', where *n* is an INTEGER greater than zero; or '*p* PERCENT', where *p* is a floating point number between 0.0 and 100.0 inclusive.

Result set

ttOptEstimateStats returns no results.

Examples

```
CALL ttOptEstimateStats ( 'ACCTS', 1, '5 PERCENT' );

CALL ttOptEstimateStats ( 'ACCTS', 1, '75 ROWS' );
```

Notes

The TimesTen statistics include the number of rows in each table, the number of unique values in each column, and the minimum and maximum values in each column. TimesTen assumes a uniform distribution of column values.

This procedure only runs faster than `ttOptUpdateStats` when you sample less than 50 percent of the rows in the table.

Estimates are not computed on columns that are longer than 2,048 bytes, and statistics for these columns are not updated. To update statistics on columns longer than 2,048 bytes, use the `ttOptUpdateStats` built-in procedure. (For varying length columns, this procedure updates statistics only if the column has a maximum length of 2,048 bytes or less.)

If a very small value is chosen for the *`sampleStr`* parameter, this procedure runs quickly but may result in suboptimal execution plans. For "good" distributions of data, a 10 percent selection is a good choice for computing statistics quickly without sacrificing plan accuracy. If the number of rows specified is sufficiently large or the table in question is sufficiently small, to improve performance TimesTen computes exact statistics anyway on all columns that have a length of 2,048 bytes or less. For example, the only difference between

```
ttOptEstimateStats ('ACCTS', 1, '100 PERCENT' )
```

and

```
ttOptUpdateStats( 'ACCTS', 1 )
```

is that the former does not compute statistics for long columns.

The statistics are stored in the `TBL_STATS` and `COL_STATS` system tables.

For performance reasons, TimesTen does not hold a lock on tables or rows when computing statistics. However, computing statistics can still slow performance. Estimating statistics generally provides better performance than computing exact statistics.

See also

`ttOptSetColIntvlStats`
`ttOptSetFlag`
`ttOptSetOrder`
`ttOptSetTblStats`
`ttOptUpdateStats`
`ttPLSQLMemoryStats`

ttOptGetColStats

Description

This procedure returns statistics information in text format.

Required privilege

This procedure requires the `SELECT` privilege on the specified tables.

Syntax

```
ttOptGetColStats('tblName', 'colName')
```

Parameters

`ttOptGetColStats` has these parameters:

Parameter	Type	Description
tblName	`TT_CHAR (61)`	Name of the table whose statistics are to be returned. If `NULL` is passed, then values for all tables are returned. Using a synonym to specify a table name is not supported.
colName	`TT_CHAR (30)`	Name of the column for which statistics should be returned. If `NULL` is passed, statistics for all columns in the specified table are returned.

Result set

`ttOptGetColStats` returns the result set:

Column	Type	Description
tblName	`TT_CHAR (30)`	Name of the table. Using a synonym to specify a table name is not supported.
colName	`TT_CHAR (30)`	Name of the column.
stats	`TT_VARCHAR (409600) NOT NULL`	Statistics in text form.

Examples

```
CALL ttOptGetColStats ();
< T1 , X1, (2, 10, 10, 100 (,4, 40, 10 ,1, 10, 5) ,(4, 20, 20 ,11, 20, 15) )>
```

See also

`ttOptSetColStats`
`ttOptSetColIntvlStats`

ttOptGetFlag

Description

This procedure returns the optimizer flag settings for the current transaction. The results are returned as a result set that can be retrieved using the ODBC `SQLFetch` function or the JDBC `ResultSet.getXXX` method, just like the result of a SQL `SELECT` statement. Applications can request the value of a specific optimizer flag by passing the flag name to `ttOptGetFlag`. Alternatively, applications can request the values of all the optimizer flags by passing `NULL`. The optimizer flags and their meanings are described under the `ttOptSetFlag` built-in procedure.

Required privilege

This procedure requires no privilege.

Syntax

```
ttOptGetFlag('flagName')
```

Parameters

`ttOptGetFlag` has the parameter:

Parameter	Type	Description
flagName	`TT_CHAR (32)`	Name of the flag whose value is to be returned. If `NULL` is passed, then the values of all flags are returned.

Result set

`ttOptGetFlag` returns the result set:

Column	Type	Description
flagName	`TT_VARCHAR (32) NOT NULL`	Name of the flag. See "ttOptSetFlag" on page 2-94 for a description of possible flag values.
value	`TT_INTEGER NOT NULL`	Current flag value, either 0 or 1.

Examples

```
CALL ttOptGetFlag('TmpHash');
```

See also

`ttOptSetFlag`

ttOptGetMaxCmdFreeListCnt

Description

This procedure returns the size of the free list of SQL compiled command cache. To reset the size of the cache, use `ttOptSetMaxPriCmdFreeListCnt` for materialized views and `ttOptSetMaxCmdFreeListCnt` for regular tables.

Required privilege

This procedure requires no privilege.

Parameters

`ttOptGetMaxCmdFreeListCnt` has no parameters.

Syntax

```
ttOptGetMaxCmdFreeListCnt()
```

Result set

`ttOptGetMaxCmdFreeListCnt` returns the results.

Column	Type	Description
retVal	`TT_VARCHAR (200) NOT NULL`	The size of the SQL compiled command cache.

Example

```
CALL ttOptGetMaxCmdFreeListCnt( );
```

See also

`ttOptSetMaxPriCmdFreeListCnt`
`ttOptSetMaxCmdFreeListCnt`

ttOptGetOrder

Description

This procedure returns a single-row result set containing the join order for the current transaction. This result set can be retrieved using the ODBC `SQLFetch` function or the JDBC `ResultSet.getXXX` method, just like the result of a SQL `SELECT` statement. Join orders are described under the `ttOptSetOrder` built-in procedure.

Required privilege

This procedure requires no privilege.

Syntax

```
ttOptGetOrder( )
```

Parameters

`ttOptGetOrder` has no parameters.

Result set

`ttOptGetOrder` returns the result set:

Column	Type	Description
joinOrder	`TT_VARCHAR(1024) NOT NULL`	Optimizer join order for the current transaction.

Examples

```
CALL ttOptGetOrder;
```

See also

`ttOptSetOrder`

ttOptSetColIntvlStats

Description

This procedure modifies the statistics for the specified columns with interval information. This procedure allows an application to set statistics manually rather than have TimesTen automatically compute them. This feature is useful for preparing commands before the data has been inserted or for seeing how table characteristics can affect the choice of execution plan. This procedure modifies the relevant row(s) in the COL_STATS system table. Modifying interval statistics for a column that is not currently indexed has no effect.

Because this procedure can be used before any data are in the table, the values specified do not need to bear any relation to the actual values, although some basic validity checking is performed.

Required privilege

This procedure requires no privilege (if owner) or `ALTER ANY TABLE` privilege (if not owner).

Syntax

```
ttOptSetColIntvlStats('tblName', 'colName', invalidate, (stats))
```

Parameters

`ttOptSetColIntvlStats` has these parameters:

Parameter	Type	Description
tblName	`TT_CHAR(61) NOT NULL`	Name of an application table. Can include table owner. Using a synonym to specify a table name is not supported.
colName	`TT_CHAR(30) NOT NULL`	Name of a column in that table.
invalidate	`TT_INTEGER`	0 (no) or 1 (yes). If *invalidate* is 1, all commands that reference the affected tables are automatically prepared again when re-executed. This includes commands prepared by other users. If *invalidate* is 0, the statistics are not considered to have been modified and existing commands are not reprepared.

Parameter	Type	Description

stats	`VARBINARY (409600) NOT NULL`	Sets stats for the column, using the format:
		`(numInterval integer, numNull integer, totUniq integer, totTups integer,`
		`/* information for interval 1 */`
		`(numUniq integer, numTups integer, frequency of most occurred value integer, minVal, maxVal, modalVal),`
		`/* information for interval 2 */...)`
		The modal value (`modalVal`) is the value that occurs most often in a specified interval.

Result set

`ttOptSetColIntvlStats` returns no results.

Example

To set the following statistics for column `t1.x1`:

- Two intervals
- Integer type
- 10 rows with null value
- 10 unique value
- 100 rows
- Interval 1 (4 unique values besides the most frequently occurring value, 40 rows with values other than most frequently occurring value, 10 rows with most frequently occurring value, min = 1, max = 10, mod = 5)
- Interval 2 (4 unique values besides the most frequently occurring value, 20 rows with values other than most frequently occurring, 20 rows with most frequently occurring value, min = 11, max = 20, mod = 15)

Use the statement:

```
CALL ttOptSetColIntvlStats('t1', 'x1', 1, (2, 10, 10, 100, (4, 40, 10, 1, 10, 5),
 (4, 20, 20, 11, 20, 15)));
```

Notes

You must specify the minimum and maximum values in the interval as `VARBINARY`. `NULL` values are not permitted as minimum or maximum values. The value is stored in the platform-specific endian format.

See also

`ttOptEstimateStats`

ttOptGetColStats
ttOptSetColStats
ttOptSetTblStats
ttOptUpdateStats

ttOptSetColStats

Description

This procedure modifies the statistics for the specified columns. This procedure allows an application to set statistics manually rather than have TimesTen automatically compute them. This feature is useful for preparing commands before the data has been inserted or for seeing how table characteristics can affect the choice of execution plan. This procedure modifies the relevant row(s) in the `COL_STATS` system table.

Because this procedure can be used before the table is populated with data, the values specified do not need to bear any relation to the actual values, although some basic validity checking is performed.

Required privilege

This procedure requires no privilege (if owner) or `ALTER ANY TABLE` privilege (if not owner).

Syntax

```
ttOptSetColStats('tblName', 'colName', numUniq, minVal,maxVal,
 invalidate, numNull)
```

Parameters

`ttOptSetColStats` has these parameters:

Parameter	Type	Description
tblName	`TT_CHAR(61) NOT NULL`	Name of an application table. Can include table owner. Using a synonym to specify a table name is not supported.
colName	`TT_CHAR(30) NOT NULL`	Name of a column in that table.
numUniq	`TT_INTEGER NOT NULL`	Number of unique values in the column.
minVal	`VARBINARY(1024) NOT NULL`	Minimum value in the column (possibly truncated).
maxVal	`VARBINARY(1024) NOT NULL`	Maximum value in the column (possibly truncated).
invalidate	`TT_INTEGER`	0 (no) or 1 (yes). If *invalidate* is 1, all commands that reference the affected tables are automatically prepared again when re-executed. This includes commands prepared by other users. If *invalidate* is 0, the statistics are not considered to have been modified and existing commands are not reprepared.

Parameter	Type	Description
numNull	TT_INTEGER	Indicates the total number of NULLs in the column.

Result set

ttOptSetColStats returns no results.

Example

```
CALL ttOptSetColStats ('SALLY.ACCTS, 'BALANCE, 400, 0x00001388, 0x000186A0, 1, 0);
```

Notes

You must specify the minimum and maximum values as VARBINARY. NULL values are not permitted as minimum or maximum values. The value is stored in the platform-specific endian format.

The statistics are treated as a single interval of column values that are uniformly distributed between the minimum value and the maximum value.

See also

ttOptEstimateStats
ttOptGetColStats
ttOptSetColIntvlStats
ttOptSetTblStats
ttOptUpdateStats

ttOptSetFlag

Description

This procedure allows applications to alter the generation of execution plans by the TimesTen query optimizer. It sets flags to enable or disable the use of various access methods. The changes made by this call take effect during preparation of statements and affect all subsequent calls to the ODBC functions `SQLPrepare` and `SQLExecDirect` or the JDBC methods `Connection.prepareCall` and `Statement.execute` in the current transaction. All optimizer flags are reset to their default values when the transaction has been committed or rolled back. If optimizer flags are set while `AutoCommit` is on, they are ignored because each statement is executed within its own transaction.

Required privilege

This procedure requires no privilege.

Syntax

```
ttOptSetFlag('optFlag', optVal)
```

Parameters

`ttOptSetFlag` has these parameters:

Parameter	Type	Description
optFlag	`TT_CHAR(32) NOT NULL`	Name of optimizer flag.
optVal	`TT_INTEGER NOT NULL`	The value of the optimizer flag. The value is generally `0` (disable/disallow) or `1` (enable/allow), except as described under "Optimizer flags" below.

Optimizer flags

When setting the optimizer flags, use the following character strings, which are not case sensitive:

Flag	Description
`DynamicLoadEnable`	Enables or disables dynamic load of Oracle data to a TimesTen dynamic cache group. By default, dynamic load of Oracle data is enabled.
`DynamicLoadErrorMode`	Enables or disables dynamic load error mode. It controls output of error messages upon failure of a transparent load operation on a TimesTen dynamic cache group. Disabled by default.
`FirstRow`	Enables or disables first row optimization in a `SELECT`, `UPDATE` or `DELETE statement`. If the SQL keyword `FIRST` is used in the SQL statement, it takes precedence over this optimizer hint. The `FIRST` keyword enables first row optimization.

Flag	Description
`GenPlan`	Enables or disables the creation of entries in the `PLAN` table for the rest of the transaction. See "Generating a query plan" in *Oracle TimesTen In-Memory Database Operations Guide*.
`GlobalProcessing`	Enables or disables global execution of `SELECT` and `UNLOAD CACHE GROUP` statements in a grid. By default, these statements are executed locally. **0** - `SELECT` and `UNLOAD CACHE GROUP` statements are executed locally. **1** - `SELECT` and `UNLOAD CACHE GROUP` statements are executed globally. See *Oracle In-Memory Database Cache User's Guide* for more details.
`Hash`	Enables or disables the use of existing hash indexes in indexed table scans.
`HashGb`	Enables or disables the use of hash groups.
`MergeJoin`	Enables or disables the use of merge joins.
`NestedLoop`	Refers to a common way of joining two tables.
`PassThrough`	Temporarily changes the pass through level for IMDB Cache applications. The pass through level can be set at any time and takes effect immediately. Legal values for this flag are: **0** - SQL statements are executed only on TimesTen. (default) **1** - `INSERT`, `UPDATE` and `DELETE` statements are executed on TimesTen unless they reference one or more tables that are not in TimesTen. If they reference one or more tables not in TimesTen, they are passed through to the Oracle database. DDL statements are executed on TimesTen. Other statements are passed through to the Oracle database if they generate a syntax error in TimesTen or if one or more tables referenced within the statement are not in TimesTen. **2** - `INSERT`, `UPDATE` and `DELETE` statements performed on tables in read-only cache groups or user managed cache groups with the `READONLY` cache table attribute are passed through to the Oracle database. Passthrough behavior for other cache group types is the same as `PassThrough=1`. **3** - All statements are passed through to the Oracle database for execution, except that `INSERT`, `UPDATE` and `DELETE` statements issued on cache tables in a dynamic AWT global cache group result in a TimesTen error. **4** - `SELECT` statements issues on cache tables in a dynamic AWT global cache group that do not satisfy the criteria for a dynamic load query are passed through to the Oracle database for execution. Otherwise, statements are executed in the TimesTen database. **5** - `SELECT` statements issued on cache tables in a dynamic AWT global cache group that do not satisfy the criteria for a dynamic load query are passed through to the Oracle database for execution when all committed updates on cache tables in dynamic AWT global cache groups by previous transactions within the connection have been propagated to the Oracle database. Otherwise, statements are executed in the TimesTen database.
`RowLock`	Allows or disallows the optimizer to consider using row locks.
`Scan`	Refers to full table scans.

Flag	Description
`Rowid`	Enables or disables the use of Row IDs.
`ShowJoinOrder`	Shows the join order of the tables in an optimizer scan.
`TmpHash`	Enables or disables the use of a temporary hash scan. This is an index that is created during execution for use in evaluating the statement. Though index creation is time-consuming, it can save time when evaluating join predicates.
`TblLock`	Enables or disables the optimizer to consider using table locks.
`TmpTable`	Stores intermediate results into a temporary table. This operation is sometimes chosen to avoid repeated evaluation of predicates in join queries or sometimes just to allow faster scans of intermediate results in joins.
`TmpTtree`	Performs a temporary range scan. Can also be used so that values are sorted for a merge join. Though index creation is time-consuming, it can save time when evaluating join predicates.
`Ttree`	Enables or disables the use of existing range indexes in indexed table scans.

In addition, the string `AllFlags` can be used to refer to all optimizer flags, and the string `Default` can be used to refer to the default flags. `Default` excludes the `GenPlan` flag but includes all other optimizer flags.

Flag description

The value of each flag can be 1 or 0:

- If 1, the operation is enabled.
- If 0, the operation is disabled unless absolutely necessary.
- Initially, all the flag values *except*`GenPlan` are 1 (all operations are permitted).

For example, an application can prevent the optimizer from choosing a plan that stores intermediate results:

```
ttOptSetFlag ( 'TmpTable', 0 )
```

Similarly, an application can specify a preference for `MergeJoin`:

```
ttOptSetFlag ( 'NestedLoop', 0 )
```

In the second example, the optimizer may still choose a nested loop join if a merge join is impossible (for example, if there is no merge-join predicate). Similarly, the optimizer may occasionally not be able to satisfy an application request to avoid table scans (when the `Scan` flag is set to `0`).

You cannot specify that a particular operation is prohibited only at a certain step of a plan or that a particular join method always be done between two specific tables. Similarly, there is no way to specify that certain indexes be used or that a hash index be used to evaluate a specific predicate. Each operation is either fully permitted or fully restricted.

When a command is prepared, the current optimizer flags, index hints and join order are maintained in the structure of the compiled form of the command and are used if the command is ever reprepared by the system. See "The TimesTen Query Optimizer" in *Oracle TimesTen In-Memory Database Operations Guide* for an example of reprepared statements.

If both `RowLock` and `TblLock` are disabled, TimesTen uses row-locking. If both `RowLock` and `TblLock` are enabled, TimesTen uses the locking scheme that is most likely to have better performance:

TblLock status	RowLock status	Effect on the optimizer
Disabled	Disabled	Use row-level locking.
Enabled	Disabled	Use table-level locking.
Disabled	Enabled	Use row-level locking.
Enabled	Enabled	Optimizer chooses row-level or table-level locking.

In general, table-level locking is useful when a query accesses a significant portion of the rows of a table and/or when there are very few concurrent transactions accessing the table.

Result set

`ttOptSetFlag` returns no results.

Example

```
CALL ttOptSetFlag ('TmpHash', 1);
```

See also

`ttOptEstimateStats`
`ttOptGetFlag`
`ttOptGetOrder`
`ttOptSetColIntvlStats`
`ttOptSetOrder`
`ttOptSetTblStats`
`ttOptUpdateStats`
`ttPLSQLMemoryStats`

ttOptSetMaxCmdFreeListCnt

Description

This procedure sets the maximum count of the free list of SQL compiled commands for regular tables. To get the current setting use the `ttOptGetMaxCmdFreeListCnt` procedure.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttOptSetMaxCmdFreeListCnt(maxCnt)
```

Parameters

`ttOptSetMaxCmdFreeListCnt` has the required parameter:

Parameter	Type	Description
maxCnt	`TT_INTEGER NOT NULL`	The max number of free SQL compiled commands for regular tables.

Result set

`ttOptSetMaxCmdFreeListCnt` returns no results.

Example

```
CALL ttOptSetMaxCmdFreeListCnt(40);
```

See also

`ttOptGetMaxCmdFreeListCnt`

ttOptSetMaxPriCmdFreeListCnt

Description

This procedure sets the maximum count of the free list of SQL compiled commands that perform materialized view maintenance.

When this command is set, freeable materialized view compiled commands are counted separately from those of regular tables. If this command is not set, materialized view compiled commands are counted as regular commands.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttOptSetMaxCmdPriFreeListCnt(maxCnt)
```

Parameters

`ttOptSetMaxPriCmdFreeListCnt` has the required parameter:

Parameter	Type	Description
maxCnt	`TT_INTEGER NOT NULL`	The size of the SQL compiled command cache.

Result set

`ttOptSetMaxPriCmdFreeListCnt` returns no results.

Example

```
CALL ttOptSetMaxPriCmdFreeListCnt(40);
```

See also

`ttOptGetMaxCmdFreeListCnt`
`ttOptSetMaxCmdFreeListCnt`

ttOptSetOrder

Description

This procedure specifies the order in which tables should be joined by the optimizer. The character string is a list of correlation names referenced in the query or a subquery, separated by spaces (*not* commas). The table listed first is scanned first by the plan. (It is outermost in a nested loop join, for example.) A correlation name is a shortcut or alias for a qualified table name.

Required privilege

This procedure requires no privilege.

Syntax

```
ttOptSetOrder('joinOrder')
```

Parameters

`ttOptSetOrder` has the required parameter:

Parameter	Type	Description
joinOrder	`TT_VARCHAR(1024)`	List of space-separated table correlation names. If an owner is required to distinguish the table name, use a table correlation name. If the *joinOrder* is not specified the query optimizer reverts to its default behavior.

Result set

`ttOptSetOrder` returns no results.

Examples

```
CALL ttOptSetOrder ('EMPS DEPTS ACCTS');
```

Use the correlation name instead of the actual table name when specifying the join order.

If an application makes the call:

```
call ttOptSetOrder('ORDERS CUSTOMERS');
```

the optimizer scans the ORDERS table before scanning the CUSTOMERS when evaluating the following query that lists all the customers who have at least one unshipped order:

```
SELECT CUSTOMERS.NAME
FROM  CUSTOMERS
WHERE EXISTS (SELECT 1
        FROM  ORDERS
        WHERE CUSTOMERS.ID = ORDERS.CUSTID
        AND  ORDER.STATUS ='UN-SHIPPED');
```

If an application makes the call:

```
ttOptSetOrder('DEPTS EMPS ACCTS');
```

the optimizer is prevented from executing a join between DEPTS and ACCTS when evaluating the number of employees working on a specific account:

```
SELECT COUNT(DISTINCT EMPS.ID)
FROM  ACCTS, DEPTS, EMPS
WHERE ACCTS.DEPTS = DEPTS.ID
AND  EMPS.DEPTS = DEPTS.ID
AND  ACCTS.NUM = :AcctNum
```

If the application does not reset the join order and tries to prepare a command that does not reference each of the three tables (and no others), the optimizer issues warning number 965. The specified join order is not applicable. TimesTen considers valid join orders and ignores the specified join order when preparing the command.

Notes

The string length is limited to 1,024 bytes. If a string exceeds this length, it is truncated and a warning is issued.

When correlation names referenced in subqueries are in included in the order, TimesTen may internally change the isolation mode.

When a command is prepared, the current optimizer flags, index hints, and join order are maintained in the structure of the compiled form of the command and are used if the command is ever reprepared by the system. See "The TimesTen Query Optimizer" in *Oracle TimesTen In-Memory Database Operations Guide* for an example of reprepared statements.

The changes made by this call take effect immediately and affect all subsequent calls to the ODBC function `SQLPrepare` or the JDBC method `Connection.prepareCall` in the current transaction. The query optimizer reverts to its default behavior for subsequent transactions.

The tables referenced by a query must exactly match the names given if the join order is to be used (the comparisons are not case sensitive). A complete ordering must be specified; there is no mechanism for specifying partial orders. If the query has a subquery then the join order should also reference the correlation names in the subquery. In essence, the join order should reference all the correlation names referenced in the query. The TimesTen optimizer internally implements a subquery as a special kind of join query with a GROUP BY. For the join order to be applicable it should reference all the correlation names. If there is a discrepancy, a warning is issued and the specified join order is ignored completely.

See also

`ttOptEstimateStats`
`ttOptGetFlag`
`ttOptGetOrder`
`ttOptSetColIntvlStats`
`ttOptSetFlag`
`ttOptSetTblStats`
`ttOptUpdateStats`
`ttPLSQLMemoryStats`

ttOptSetTblStats

Description

This procedure modifies the statistics for the specified table. This procedure allows an application to set statistics explicitly rather than have TimesTen automatically compute them.

Required privilege

This procedure requires no privilege (if owner) or `ALTER ANY TABLE` privilege (if not owner).

Syntax

```
ttOptSetTblStats('tblName', numRows, invalidate)
```

Parameters

`ttOptSetTblStats` has these parameters:

Parameter	Type	Description
tblName	`TT_CHAR(61) NOT NULL`	Name of an application table. Can include table owner. Using a synonym to specify a table name is not supported.
numRows	`TT_INTEGER NOT NULL`	Number of rows in the table.
invalidate	`TT_INTEGER`	0 (no) or 1 (yes). If *invalidate* is 1, all commands that reference the affected tables are automatically prepared again when re-executed, including commands prepared by other users. If *invalidate* is 0, the statistics are not considered to have been modified and existing commands are not reprepared.

Result set

`ttOptSetTblStats` returns no results.

Example

```
CALL ttOptSetTblStats ( 'ACCTS', 10000, 0 );
```

Note

This feature is useful for preparing commands before the data has been inserted or for seeing how table size can affect the choice of an execution plan. Because the command can be used before any data are in the table, the values specified do not need to bear any relation to the actual values. This procedure modifies the relevant row(s) in the `TBL_STATS` system table. See "`SYS.TBL_STATS`" in *Oracle TimesTen In-Memory Database System Tables and Limits Reference.*

See also

ttOptEstimateStats
ttOptGetFlag
ttOptGetOrder
ttOptSetColIntvlStats
ttOptSetFlag
ttOptSetOrder
ttOptUpdateStats
ttPLSQLMemoryStats

ttOptShowJoinOrder

Description

This procedure returns the join order of the last prepared or executed SQL statement (`SELECT`, `UPDATE`, `DELETE`, and `INSERT SELECT`) in the current transaction. For a join order to be collected, use `ttOptSetFlag('ShowJoinOrder', 1)` or set the ttIsql "ShowJoinOrder" command to `ON` (1) first in the same transaction. `AUTOCOMMIT` must be off when using either of these commands. The join order is represented by table names.

Required privilege

This procedure requires no privilege.

Syntax

```
ttOptShowJoinOrder()
```

Parameters

`ttOptShowJoinOrder` has no parameters.

Result set

`ttOptShowJoinOrder` returns the result:

Column	Type	Description
tblName	`TT VARCHAR (4096) NOT NULL`	Table names, including owner name quantifiers and correlation name for each table if specified. Table names are returned in parentheses. Using a synonym to specify a table name is not supported.

Example

```
>AUTOCOMMIT 0;
> CALL ttOptSetFlag ('ShowJoinOrder', 1);
>PREPARE SELECT * FROM t1;
>CALL ttOptShowJoinOrder();
>( T1 )
```

Notes

You must call `ttOptSetFlag('ShowJoinOrder', 1)` or set the `ttIsql` "ShowJoinOrder" command to ON (1) before using this procedure.

This procedure works within one transaction and is not persistent across transactions.

See also

`ttOptEstimateStats`
`ttOptGetFlag`
`ttOptGetOrder`
`ttOptSetColIntvlStats`

ttOptSetFlag
ttOptSetOrder
ttOptSetTblStats
ttOptUpdateStats
ttPLSQLMemoryStats

ttOptUpdateStats

Description

This procedure updates the statistics for the specified table. TimesTen looks at the databases in the table and updates the TBL_STATS and COL_STATS system tables. If the table is large, this process can take some time. Statistics are not computed automatically as rows are updated; an application must compute them explicitly by calling this procedure.

Required privilege

This procedure requires no privilege for the table owner. This procedure requires no privilege if *tblName* is not specified, because the procedure operates on the current user's tables if *tblName* is not specified.

This procedure requires the `ALTER ANY TABLE` privilege if user is not the table owner.

Syntax

```
ttOptUpdateStats('tblName', invalidate, option)
```

Parameters

`ttOptUpdateStats` has these parameters:

Parameter	Type	Description
tblName	`TT_CHAR(61)`	Name of an application table. Can include table owner. If a value of `NULL` or an empty string is provided, the statistics for all the current user's tables are updated. Using a synonym to specify a table name is not supported.
invalidate	`TT_INTEGER`	0 (no) or 1 (yes). If *invalidate* is 1, all commands that reference the affected tables are automatically prepared again when re-executed, including commands prepared by other users. If *invalidate* is 0, the statistics are not considered to have been modified and existing commands are not reprepared. The *invalidate* parameter is optional and defaults to 0.

Parameter	Type	Description
option	TT_INTEGER	Specifies whether to collect complete interval statistics information. Valid values for this option are: Null or 0 - Collect complete interval statistics only if a range index exists on the column. If a range index does not exist, only single interval statistics are collected. 1 - Do not collect complete interval statistics. Only single interval statistics are collected. See "Notes" for more information.

Result set

`ttOptUpdateStats` returns no results.

Example

```
CALL ttOptUpdateStats ( 'ACCTS', 1 );
```

Updates the ACCTS table and causes all commands that reference the ACCTS table to be re-prepared when they are next executed.

```
CALL ttOptUpdateStats('', 1);
```

Updates all the current user's tables and causes commands on those tables to be reprepared when they are next executed.

```
CALL ttOptUpdateStats('ACCTS', 0, 1);
```

Forces single interval statistics to be collected.

Notes

If the table name specified is an empty string, statistics are updated for all the current user's tables.

When complete interval statistics are collected, the total number of rows in the table is divided into 20 or less intervals and the distribution of each interval is recorded in the statistics. The new statistics contain the information:

- Number of intervals
- Total number of `NULL`s in the column
- Total number of `NON NULL UNIQUE` values in the column
- Total number of rows in the table
- Interval information, each interval contains:
- The minimum value
- The maximum value
- The most frequently occurring value
- The number of times the most frequent value occurred

- The number of rows that have different values than the most frequent value
- The number of unique values besides the most frequent value

Collection of complete interval statistics requires the data to be sorted.

If complete interval statistics are not selected, then statistics are collected by treating the entire distribution as a single interval.

For performance reasons, TimesTen does not hold a lock on tables or rows when computing statistics. However, computing statistics can still slow performance. Estimating statistics generally provides better performance than computing exact statistics. See "ttOptEstimateStats" on page 2-83 for information on estimating statistics.

See also

`ttOptEstimateStats`
`ttOptGetColStats`
`ttOptSetColStats`
`ttOptSetColIntvlStats`
`ttOptSetTblStats`
`ttOptUpdateStats`

ttOptUseIndex

Description

This procedure allows applications to alter the generation of execution plans by the TimesTen query optimizer. It allows applications to disable the use of a set of indexes or enable the consideration of only a set of indexes for each correlation used in a query. Enabling the consideration of an index does not guarantee that the plan generated uses the index. Depending on the estimated cost, the optimizer might choose to use a serialization scan or a materialization scan to access the associated correlation if these scans resulted in a better plan than the ones that use the specified index.

The changes made by this call take effect immediately and affect all subsequent calls to the ODBC functions `SQLPrepare` and `SQLExecDirect` or the JDBC methods `Connection.prepareCall` and `Statement.execute` in the current transaction until the applications explicitly issue a call to clear it. The setting is cleared whenever a new transaction is started.

Required privilege

This procedure requires no privilege.

Syntax

```
ttOptUseIndex('IndexName, CorrelationName, 0 | 1 [;...]')
```

Parameters

`ttOptUseIndex` has a single string parameter, *`indOption`*, of type `TT_VARCHAR(1024)` with these components:

Component	Description
`IndexName`	The name of the user-defined index or '_TMPTTREE' for temporary range index or '_TMPHASH' for temporary hash index. If index name is omitted, the setting applies to all indexes of the specified correlation.
`CorrelationName`	The correlation name of the table. If a table is defined with a correlation name in the FROM clause, use this correlation name instead of the table name when specifying the index hint for this table. If correlation name is omitted for an entry, the setting affects all tables with the specified index name.
`0 | 1`	Disables(`0`) or enables (`1`) the use of the index specified by *`IndexName`*.

Result set

`ttOptUseIndex` returns no results.

Examples

```
CALL ttOptUseIndex('"3456"."1234", t1, 0');

CALL ttOptUseIndex('data1.i1, data1.t1, 0');

CALL ttOptUseIndex('i1, t1, 0');
```

Note

If `ttOptUseIndex` is called without a parameter or with a `NULL` value, TimesTen clears the previous index hint.

See also

`ttOptEstimateStats`
`ttOptGetFlag`
`ttOptGetOrder`
`ttOptSetColIntvlStats`
`ttOptSetFlag`
`ttOptSetOrder`
`ttOptSetTblStats`
`ttOptUpdateStats`
`ttPLSQLMemoryStats`

ttPLSQLMemoryStats

Description

This procedure returns result statistics about PL/SQL library cache performance and activity.

Required privilege

This procedure requires no privilege.

Syntax

```
ttPLSQLMemoryStats(paramName, paramValue )
```

Parameters

`ttPLSQLMemoryStats` takes no parameters.

Parameters

`ttPLSQLMemoryStats` returns the results in the following columns:

Columns	Type	Description
paramName	`TT_VARCHAR(30) NOT NULL`	The name of the result statistic returned in this row.
paramValue	`BINARY_FLOAT NOT NULL`	The value of the result statistic returned in this row.

The following statistics are returned:

- `Gets`: Number of times a lock was requested for a PL/SQL object.
- `GetHits`: Number of times a PL/SQL object's handle was found in memory.
- `GetHitRatio`: Ratio of GetHits to Gets.
- `Pins`: Number of times a PIN was requested for PL/SQL objects.
- `PinHits`: Number of times all the metadata pieces of the library object were found in memory.
- `PinHitRatio`: Ratio of PinHits to Pins.
- `Reloads`: Any PIN of an object that is not the first PIN performed since the object handle was created, and which requires loading the object from the database.
- `Invalidations`: Total number of times objects in this namespace were marked invalid because a dependent object was modified.
- `CurrentConnectionMemory`: The total amount of heap memory, in Megabytes, allocated to PL/SQL on this database connection.
- `DeferredCleanups`: Total number of times a deferred cleanup occurred.

Examples

```
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/scratch/timesten/sample;
```

```
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;PermSize=128;
TypeMode=0;PLSQL_MEMORY_SIZE=32;PLSQL_MEMORY_ADDRESS=20000000;PLSQL=1;
(Default setting AutoCommit=1)
Command> create procedure hello is begin dbms_output.put_line('Hello, World!');
end;
    > /
Procedure created.
Command> call ttPlsqlMemoryStats;
< Gets, 485.00000 >
< GetHits, 444.000000 >
< GetHitRatio, .9154639 >
< Pins, 260.00000 >
< PinHits, 178.000000 >
< PinHitRatio, .6846154 >
< Reloads, 4.000000 >
< Invalidations, 0.000000e+00 >
< CurrentConnectionMemory, 56.00000 >
9 rows found.
```

ttRamPolicyGet

Description

This procedure returns the RAM policy used to determine when a database is loaded into memory. The policy can be either `always`, `manual`, or `inUse`.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRamPolicyGet()
```

Result set

`ttRamPolicyGet` returns the results:

Column	Type	Description
ramPolicy	`TT_VARCHAR (10)`	The policy used to determine when the database is loaded into system RAM. Valid values are: `always` - specifies that the database should remain in system RAM all the time. `manual` - specifies that the database is only to be loaded in system RAM when explicitly loaded by the user, using the `ttAdmin -ramLoad` command. `inUse` - specifies that the database is only loaded in system RAM when in use (i.e., when applications are connected). This option cannot be used with temporary databases. TimesTen only allows a temporary database to be loaded into RAM manually. Trying to set the policy generates a warning.
ramGrace	`TT_INTEGER`	If the *ramPolicy* is `inUse`, this field reports the number of seconds the database is kept in RAM after the last application has disconnected. Otherwise, this field is `NULL`.

Parameters

`ttRamPolicyGet` has no parameters.

Examples

To view the RAM policy, use:

```
CALL ttRamPolicyGet();
```

See also

ttAdmin
ttRamPolicySet
"Specifying a RAM policy" in *Oracle TimesTen In-Memory Database Operations Guide*

ttRamPolicySet

Description

This procedure defines the policy used to determine when a database is loaded into memory. The policy can be either `always`, `manual`, or `inUse`.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRamPolicySet('ramPolicy', ramGrace)
```

Parameters

`ttRamPolicySet` has the parameters:

Parameter	Type	Description
ramPolicy	`TT_VARCHAR (10) NOT NULL`	The policy used to determine when the database is loaded into system RAM. Valid values are: `always` - specifies that the database should remain in system RAM all the time. `manual` - specifies that the database is only to be loaded in system RAM when explicitly loaded by the user, using the `ttAdmin -ramLoad` command. `inUse` - specifies that the database is only loaded in system RAM when in use (i.e.:when applications are connected). This option cannot be used with temporary databases. TimesTen only allows a temporary database to be loaded into RAM manually. Trying to set the policy generates a warning.
ramGrace	`TT_INTEGER`	Sets the number of seconds the database is kept in RAM after the last application has disconnected. This number is only effective if *ramPoliy* is `inUse`. This parameter is optional, and when omitted or set to `NULL`, the existing *ramGrace* period is left unchanged.

Result set

`ttRamPolicySet` returns no results.

Examples

To set the policy for loading a database into RAM to be inUse and for the database to kept in RAM for 10 seconds after the last application has disconnected, use:

```
CALL ttRamPolicySet('inUse', 10);
```

See also

`ttAdmin`
`ttRamPolicyGet`

"Specifying a RAM policy" in *Oracle TimesTen In-Memory Database Operations Guide*

ttRedundantIndexCheck

Description

This procedure scans the indicated table (or all the current user's tables) to find redundant indexes. It returns the names of the redundant indexes and a suggestion for which to drop.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRedundantIndexCheck('tblname')
```

Parameters

`ttRedundantIndexCheck` has the parameter:

Parameter	Type	Description
tblName	`TT_CHAR(61)`	Name of an application table. Can include table owner. If a value of `NULL` or an empty string is provided, the redundant indexes for all the current user's tables. Using a synonym to specify a table name is not supported.

Result Set

`ttRedundantIndexCheck` returns the result:

Column	Type	Description
redundancy	`TT_VARCHAR (1024) NOT NULL`	The names of redundant indexes and a suggestion for which index to drop.

Example

Create table `y` with a primary key. Then create index `i`. TimesTen returns a warning that a redundant index is being created. Create another index, `i1`. The command fails and TimesTen returns an error. Call this procedure to show the warnings.

```
CREATE TABLE y (ID tt_integer primary key);
CREATE INDEX i ON y (id);

Warning 2240: New non-unique index I has the same key columns as existing
unique index Y; consider dropping index I

CREATE INDEX i1 ON y (id);

2231: New index I1 would be identical to existing index I The command failed.

CALL ttredundantindexcheck ('y');

< Non-unique index SCOTT.Y.I has the same key columns as unique index SCOTT.Y.Y;
consider dropping index SCOTT.Y.I >
```

```
1 row found.
```

ttRepDeactivate

Description

This procedure changes the state of the active database in an active standby pair from `ACTIVE` to `IDLE`. Use this procedure when reversing the roles of the master databases in an active standby pair.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepDeactivate()
```

Parameters

`ttRepDeactivate` has no parameters.

Result set

`ttRepDeactivate` returns no results.

Example

To deactivate the active database in an active standby pair, use:

```
CALL ttRepDeactivate();
```

See also

`ttRepTransmitGet`
`ttRepTransmitSet`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStateSave`
`ttRepStateSet`
`ttRepStop`
`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttReplicationStatus

Description

This procedure returns the status of one or more replication peer databases.

Required privilege

This procedure requires no privilege.

Syntax

```
ttReplicationStatus('receiver', 'hostname')
```

Parameters

`ttReplicationStatus` has the optional parameters:

Parameter	Type	Description
receiver	`TT_VARCHAR`(200)	Subscriber of interest or `NULL` for all subscribers. If the parameter is provided, then it names a replication subscriber about which information is sought. If the parameter is not provided, then information on replication subscribers defined for the current database is returned.
hostname	`TT_VARCHAR`(200)	The host name of one or more stores that are configured to receive updates from the executing store; if `NULL`, then receiving stores are identified by subscriber alone. If both receiver and hostname are `NULL`, then all receiving stores are selected.

Result set

`ttReplicationStatus` returns the result set:

Column	Type	Description
subscriber	`TT_VARCHAR(200) NOT NULL`	Subscriber name.
hostName	`TT_VARCHAR(200) NOT NULL`	Host name.
port	`TT_INTEGER NOT NULL`	Defined port number.

Column	Type	Description
pState	TT_CHAR(10) NOT NULL	Peer state. The values of the result column are: start - replication is enabled to this peer. pause - replication is temporarily paused to this peer. stop - replication updates are NOT being collected for this peer. failed - replication to a subscriber is considered failed because the threshold limit (log data) has been exceeded. This state is set by the system.
logs	TT_INTEGER NOT NULL	Number of transaction log files held for this peer.
lastMsg	TT_INTEGER	Seconds since last interaction or NULL.
replicationName	TT_CHAR(30) NOT NULL	Name of replication scheme.
replicationOwner	TT_CHAR(30) NOT NULL	Owner of replication scheme.

Example

```
CALL ttReplicationStatus('System8');
```

Notes

If the *receiver* parameter is not NULL, only the status of the given receiver is returned. If the *receiver* parameter is NULL, the status of all subscribers is returned.

This procedure is supported only for TimesTen Data Manager ODBC applications. It is not supported for TimesTen Client or JDBC applications.

See also

ttRepDeactivate
ttRepPolicySet
ttRepStop
ttRepSubscriberStateSet
ttRepSyncGet
ttRepSyncSet
ttRepTransmitSet
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepPolicyGet

Description

This procedure returns the replication restart policy used to determine when the TimesTen Replication Agent for the connected database should run. The policy can be `always`, `manual`, or `norestart`.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRepPolicyGet()
```

Parameters

`ttRepPolicyGet` has no parameters.

Result set

`ttRepPolicyGet` returns the results:

Parameter	Type	Description
repPolicy	`TT_VARCHAR (10)`	The policy used to determine when the TimesTen Replication Agent for the database should run. Valid values are: `always` - specifies that the replication agent for the database is always running. This option immediately starts the TimesTen Replication Agent. When the TimesTen daemon restarts, TimesTen automatically restarts the Replication Agent. `manual` - specifies that you must manually start the Replication Agent using either the `ttRepStart` built-in procedure or the `ttAdmin -repStart` command. You must explicitly stop the Replication Agent using either the `ttRepStop` built-in procedure or the `ttAdmin -repStop` command. `norestart` - specifies that the replication agent for the database is not to be restarted after a failure.

Examples

To set the policy for TimesTen Replication Agent to always, use:

```
CALL ttRepPolicyGet();
```

See also

`ttRepDeactivate`
`ttRepTransmitSet`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStart`

ttRepStop
ttRepSubscriberStateSet
ttRepSubscriberWait
ttRepSyncGet
ttRepSyncSet
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepPolicySet

Description

This procedure defines the replication restart policy used to determine when the TimesTen Replication Agent for the connected database should run. The policy can be either `always`, `manual`, or `norestart`.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepPolicySet('repPolicy')
```

Parameters

`ttRepPolicySet` has this parameter:

Parameter	Type	Description
repPolicy	`TT_VARCHAR (10) NOT NULL`	Specifies the policy used to determine when the TimesTen Replication Agent for the database should run. Valid values are: `always` - specifies that the replication agent for the database is always running. This option immediately starts the TimesTen Replication Agent. When the TimesTen daemon restarts, TimesTen automatically restarts the Replication Agent. `manual` - specifies that you must manually start the Replication Agent using either the `ttRepStart` built-in procedure or the `ttAdmin -repStart` command. You must explicitly stop the Replication Agent using either the `ttRepStop` built-in procedure or the `ttAdmin -repStop` command. `norestart` - specifies that the replication agent for the database is not to be restarted after a failure.

Result set

`ttRepPolicySet` returns no results.

Examples

To set the policy for TimesTen Replication Agent to always, use:

```
CALL ttRepPolicySet('always');
```

See also

ttRepDeactivate
ttRepTransmitSet
ttReplicationStatus
ttRepPolicyGet
ttRepStart
ttRepStop
ttRepSubscriberStateSet
ttRepSubscriberWait
ttRepSyncGet
ttRepSyncSet
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepQueryThresholdGet

Description

This procedure returns the number of seconds that was most recently specified as the query threshold for the replication agent. The number of seconds returned may not be the same as the query threshold in effect. Setting a new value for the query threshold takes effect the next time the replication agent is started.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepQueryThresholdGet()
```

Parameters

`ttRepQueryThresholdGet` has no parameters.

Result set

`ttRepQueryThresholdGet` returns the result:

Column	Type	Description
repQueryThreshold	`TT_INTEGER`	The number of seconds that a replication query executes before returning an error.

Examples

To get the replication query threshold value, use:

```
CALL ttRepQueryThresholdGet;
< 4 >
1 row found.
```

See also

`ttRepDeactivate`
`ttReplicationStatus`
`ttRepPolicyGet`
`ttRepQueryThresholdSet`
`ttRepStart`
`ttRepStop`
`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
`ttRepSyncGet`
`ttRepSyncSet`
`ttRepTransmitSet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepQueryThresholdSet

Description

This procedure specifies the number of seconds that a query can be executed by the replication agent before TimesTen writes a warning to the support log and throws an SNMP trap. The specified value takes effect the next time the replication agent is started. The query threshold for the replication agent applies to SQL execution on detail tables of materialized views, ON DELETE CASCADE operations and some internal operations that execute SQL statements.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepQueryThresholdSet(seconds);
```

Parameters

`ttRepQueryThresholdSet` has the parameter:

Parameter	Type	Description
seconds	`TT_INTEGER NOT NULL`	Number of seconds a SQL statement can be executed by the replication agent before TimesTen writes a warning to the support log and throws an SNMP trap. The value must be greater than or equal to 0. Default is 0 and indicates that Timesten does not write any warnings.

Result set

`ttRepQueryThresholdSet` returns no results.

Examples

To set the replication query threshold value to four seconds, use:

```
CALL ttRepQueryThresholdSet(4);
```

See also

ttRepDeactivate
ttReplicationStatus
ttRepPolicyGet
ttRepQueryThresholdGet
ttRepStart
ttRepStop
ttRepSubscriberStateSet
ttRepSubscriberWait
ttRepSyncGet
ttRepSyncSet
ttRepTransmitSet

"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepStart

Description

This procedure starts the TimesTen Replication Agent for the connected database.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttRepStart()
```

Parameters

`ttRepStart` has no parameters.

Result set

`ttRepStart` returns no results.

Examples

To start the replication agent, use:

```
CALL ttRepStart();
```

Notes

The replication agent does not start if the database does not participate in any replication scheme.

When using this procedure, no application, including the application making the call, can be holding a connection that specifies database-level locking (`LockLevel=1`).

See also

`ttRepDeactivate`
`ttRepTransmitGet`
`ttRepTransmitSet`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStop`
`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
`ttRepSyncSet`
`ttRepSyncGet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepStateGet

Description

This procedure returns the current replication state of a database in an active standby pair.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRepStateGet()
```

Parameters

`ttRepStateGet` has no parameters.

Result set

`ttRepStateGet` returns the result:

Column	Type	Description
`state`	`TT_VARCHAR (20) NOT NULL`	The current replication state of the database. One of: `ACTIVE` - The database is currently the active master database. Applications may update its replicated tables `STANDBY` - The database is the standby master database. Applications may only update its non-replicated tables. `FAILED` - The database is a failed master database. No updates are replicated to it. `IDLE` - The database has not yet been assigned its role in the active standby pair. It cannot be updated by applications or replication. Every store comes up in the IDLE state `RECOVERING` - The store is in the process of synchronizing updates with the active store after a failure.

Column	Type	Description
gridState	TT_VARCHAR (20) NOT NULL	The current grid state of the database. One of: NO GRID - The node is not attached. AVAILABLE - The node is attached and the role of the node is consistent with the replication store state, either active or standby. Operations that could cause ownership change can be performed. These operations include AGING, DELETE, INSERT, LOAD, SELECT, and UNLOAD. IN TRANSITION - The node is attached and is in transition to active state from standby state. This state can occur during the failover of an active standby state. The replication store is active but not available for operations that can change ownership. Operations that can change ownership are disallowed. UNAVAILABLE - The node is attached but could not be switched to active grid state during the last failover due to an error. Replication store state is already in active state, but operations that can change ownership are disallowed. The user must fix the error condition and explicitly execute the ttGridAttach procedure to bring the node to active state.

Examples

To determine the replication and the grid state of the active standby pair, use:

```
Call ttRepStateGet();
<STANDBY, NO GRID>

Call ttRepStateGet();
<ACTIVE, NO GRID>

Call ttRepStateGet();
<ACTIVE, AVAILABLE>

Call ttRepStateGet();
<ACTIVE, UNAVAILABLE>
```

See also

ttRepDeactivate
ttRepTransmitSet
ttReplicationStatus

`ttRepPolicySet`
`ttRepStateSave`
`ttRepStateSet`
`ttRepStop`
`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepStateSave

Description

This procedure saves the state of a remote peer database in an active standby pair to the currently connected database. Currently, may only be used to indicate to the active database that the standby database, *storeName* on *hostName*, has failed, and that all updates on the active database should be replicated directly to the read-only subscribers.

Required privilege

This procedure requires the ADMIN privilege.

Syntax

```
ttRepStateSave('state', 'storeName', 'hostName')
```

Parameters

ttRepStateSave has these parameters:

Parameter	Type	Description
state	TT_VARCHAR (20) NOT NULL	The replication state of the indicated database. May only be specified as FAILED in this release. Recording that a standby database has failed indicates that all replicated updates are to be sent directly from the active database to the read-only subscribers.
storeName	TT_VARCHAR (200) NOT NULL	Name of the database for which the state is indicated.
hostName	TT_VARCHAR (200)	Name of the host where the database resides.

Result set

ttRepStateSave returns no results.

Examples

To indicate to the active database that the standby database standby on host backup1 has failed, use:

```
ttRepStateSave('FAILED', 'standby', 'backup1');
```

See also

ttRepDeactivate
ttRepTransmitSet
ttReplicationStatus
ttRepPolicySet
ttRepStateGet
ttRepStateSet
ttRepStop
ttRepSubscriberStateSet

`ttRepSubscriberWait`
`ttRepSyncGet`
`ttRepSyncSet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepStateSet

Description

This procedure sets the replication state of a database in an active standby pair replication scheme. Currently, `ttRepStateSet` may only be used to set the state of a database to `ACTIVE`, indicating that it is to take the active role in an active standby pair. `ttRepStateSet` may only be executed in the following situations:

- A database has had a `CREATE ACTIVE STANDBY PAIR` command executed and no failures have occurred since.
- A database is currently in the `STANDBY` state, and the other database in the active standby pair has had its state changed from `ACTIVE` to `IDLE` using the `ttRepDeactivate` procedure.
- A database has just recovered from the local transaction log and was in the `ACTIVE` state before it went down.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepStateSet('state')
```

Parameters

`ttRepStateSet` has the parameter:

Parameter	Type	Description
state	`TT_VARCHAR (20) NOT NULL`	The replication state of the database. Must be `ACTIVE`, in this release. Setting a store to `ACTIVE` designates it as the active database in an active standby pair.

Result set

`ttRepStateSet` returns no results.

Examples

To set the replication state of the database to ACTIVE, use:

```
CALL ttRepStateSet('ACTIVE');
```

See also

`ttRepDeactivate`
`ttRepTransmitSet`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStateGet`
`ttRepStateSave`
`ttRepStop`

`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
`ttRepSyncGet`
`ttRepSyncSet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepStop

Description

This procedure stops the TimesTen replication agent for the connected database.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttRepStop()
```

Parameters

`ttRepStop` has no parameters.

Result set

`ttRepStop` returns no results.

Examples

To stop the replication agent, use:

```
CALL ttRepStop();
```

Note

When using this procedure, no application, including the application making the call, can be holding a connection that specifies database-level locking (`LockLevel=1`).

See also

`ttRepDeactivate`
`ttRepTransmitSet`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStart`
`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
`ttRepSyncGet`
`ttRepSyncSet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepSubscriberStateSet

Description

This procedure changes a replicating subscriber's state with respect to the executing master store.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepSubscriberStateSet('replicationName', 'replicationOwner',
 'subscriberStoreName', 'subscriberHostName', newStateCode)
```

Parameters

`ttRepSubscriberStateSet` has these parameters:

Parameter	Type	Description
replicationName	`TT_CHAR (30)`	The name of the replication scheme on which to operate. May be `NULL` to indicate all replication schemes.
replicationOwner	`TT_CHAR (30)`	The owner of the replication scheme. May be `NULL` to indicate all replication scheme owners.
subscriberStoreName	`TT_VARCHAR (200)`	The name of the subscribing database whose state is to be set. May be `NULL` to indicate all stores on host *subscriberHostName*.
subscriberHostName	`TT_VARCHAR (200)`	The subscriber's host. May be `NULL` to indicate all hosts of subscribing peers.
newStateCode	`TT_INTEGER`	An integer code representing the specified subscriber's new state: 0/`NULL` - started 1 - paused 2 - stopped All other state codes are disallowed. (This procedure cannot set a subscriber state to "failed.")

Result set

`ttRepSubscriberStateSet` returns no results.

Examples

For the replication scheme named `REPL.REPSCHEME`, the following directs the master database to set the state of the subscriber database (`SUBSCRIBERDS ON SYSTEM1`) to `Stop` (`2`):

```
CALL ttRepSubscriberStateSet('REPSCHEME', 'REPL', 'SUBSCRIBERDS','SYSTEM1', 2);
```

To direct the master database to set the state of all its subscribers to `Pause` (`1`), use:

```
CALL ttRepSubscriberStateSet( , , , , 1 );
```

Leaving a parameter empty is equivalent to using `NULL`.

See also

`ttRepDeactivate`
`ttRepTransmitSet`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStart`
`ttRepStop`
`ttRepSubscriberWait`
`ttRepTransmitGet`
`ttRepTransmitSet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepSubscriberSync

Description

This procedure checks a subscriber database to determine if all updates have been received from a specified master database.

This built-procedure can only be invoked from a subscriber and only one connection at a time can invoke the procedure.

When the procedure returns, either it has timed out or all transactions that committed before it was invoked have been shipped and applied on the subscriber. You must supply a value for all the arguments in this procedure.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepSubscriberSync('replicationName', 'replicationOwner', 'MASTERStoreName',
 'masterHostName', waitTime)
```

Parameters

`ttRepSubscriberSync` has these parameters:

Parameter	Type	Description
replicationName	`TT_CHAR (30) NOT NULL`	The name of the replication scheme on which to operate. May be `NULL` to indicate all replication schemes.
replicationOwner	`TT_CHAR (30) NOT NULL`	The owner of the replication scheme. May be `NULL` to indicate all replication scheme owners.
masterStoreName	`TT_VARCHAR (200) NOT NULL`	The name of the master database with which the subscriber must sync.
masterHostName	`TT_VARCHAR(200) NOT NULL`	The master host. May be `NULL` to indicate all hosts of master peers.
waitTime	`TT_INTEGER NOT NULL`	Number of seconds to wait for the specified master(s). A value of -1 indicates to wait forever. This parameter is required and may not be `NULL`.

Result Set

`ttRepSubscriberSync` returns the result set:

Column	Type	Description
timeOut	BINARY(1)	0x00 - The wait succeeded within the allotted *waitTime*; the specified subscribers are up to date at the time this procedure was called. TimesTen returns 0x01 if not enough time has been granted.

Example

If there is one defined replication scheme REPOWNER.REPSCHEME, to direct the subscriber database to wait ten minutes to catch up with master REP1 on SERVER1, use:

```
CALL ttRepSubscriberSync('REPSCHEME, 'REPOWNER, 'REP1', 'SERVER1', 600);
```

See also

ttRepDeactivate
ttRepTransmitSet
ttReplicationStatus
ttRepPolicySet
ttRepStart
ttRepStop
ttRepSubscriberStateSet
ttRepSubscriberWait
ttRepSyncGet
ttRepSyncSet
ttRepSyncSubscriberStatus
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepSubscriberWait

Description

This procedure causes the caller to wait until all transactions that committed before the call have been transmitted to the subscriber *subscriberStoreName*. It also waits until the subscriber has acknowledged that the updates have been durably committed at the subscriber database.

Call this procedure in a separate transaction, when no other transaction is pending on the active database. This call returns an error if any transactions on the active database are open.

If you set the *waitTime* parameter to -1 and the *subscriberStoreName* parameter to NULL, the ttRepSubscriberWait procedure does not return until all updates committed up until the time of the procedure call have been transmitted to all subscribers, and all subscribers have acknowledged that the updates have been durably committed.

The ttRepSubscriberWait procedure should not be used when an urgent response is required. Instead, you should use the return receipt service.

> **Note:** If this procedure is called after all write transaction activity is quiesced at a store (there are no active transactions and no transactions have started), it may take a 60 seconds or longer before the subscriber sends the acknowledgment that all updates have been durably committed at the subscriber.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRepSubscriberWait('replicationName', 'replicationOwner', 'subscriberStoreName',
 'subscriberHostName', waitTime)
```

Parameters

ttRepSubscriberWait has these parameters:

Parameter	Type	Description
replicationName	TT_CHAR (30)	The name of the replication scheme on which to operate. May be NULL to indicate all replication schemes.
replicationOwner	TT_CHAR (30)	The owner of the replication scheme. May be NULL to indicate all replication scheme owners.
subscriberStoreName	TT_VARCHAR (200)	The name of the subscribing database whose state is to be set. May be NULL to indicate all stores on host *subscriberHostName*.

Parameter	Type	Description
subscriberHostName	TT_VARCHAR(200)	The subscriber's host. May be NULL to indicate all hosts of subscribing peers.
waitTime	TT_INTEGER NOT NULL	Number of seconds to wait for the specified subscriber(s). A value of -1 indicates to wait forever. This parameter is required and may not be NULL.

Result Set

ttRepSubscriberWait returns the result set:

Column	Type	Description
timeOut	BINARY(1)	0x00 - The wait succeeded within the allotted *waitTime*; the specified subscribers are up to date at the time this procedure was called. TimesTen returns 0x01 if not enough time has been granted.

Example

If there is one defined replication scheme REPOWNER.REPSCHEME, to direct the transmitting database to wait ten minutes for subscriber REP2 on SERVER2 to catch up, use:

```
CALL ttRepSubscriberWait('REPSCHEME','REPOWNER', 'REP2', 'SERVER2', 600);
```

See also

ttRepDeactivate
ttRepTransmitSet
ttReplicationStatus
ttRepPolicySet
ttRepStart
ttRepStop
ttRepSubscriberStateSet
ttRepSyncGet
ttRepSyncSet
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepSyncGet

Description

This procedure returns static attributes associated with the caller's use of the replication- based return service. This procedure operates with either the `RETURN RECEIPT` or `RETURN TWOSAFE` service.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRepSyncGet()
```

Parameters

`ttRepSyncGet` has no parameters.

Result set

`ttRepSyncGet` returns the result set:

Column	Type	Description
requestReturn	`BINARY(1)`	0 - Don't wait for return notification configured with the `RETURN RECEIPT BY REQUEST` or `RETURN TWOSAFE BY REQUEST` option; this value is the default. 1 - Wait for the return notification. Commit resets this attribute to its default value of 0 ("off").
returnWait	`TT_INTEGER`	Specifies the number of seconds to wait for return service acknowledgment. The default value is 10 seconds. A value of `0' means that there is no wait time.This attribute persists across transaction boundaries and applies to all `RETURN` services independent of the BY `REQUEST` option.

Column	Type	Description
localAction	TT_INTEGER	The current LOCAL ACTION configuration for RETURN services. 1 -- NO ACTION -- When a COMMIT times out, it returns the application unblocked, leaving the transaction in the same state it was when the COMMIT began,. The application may only reissue the COMMIT. This is the default. 2 -- COMMIT. When the COMMIT times out, the transaction is committed locally. No more operations are possible on this transaction, and the replicated databases diverge.This attribute persists across transactions and for the life of the connection.

Example

To retrieve the caller's *requestReturn* value, use:

```
SQLCHAR requestReturn[1];
SQLINTEGER    len;
rc = SQLExecDirect   ( hstmt
                , (SQLCHAR *) "{CALL ttRepSyncGet( NULL )}"
                , SQL_NTS )
rc = SQLBindCol     ( hstmt
                , /* ColumnNumber */      1
                , /* TargetType */      SQL_C_BINARY )
                , /* TargetValuePtr */     requestReturn
                ,./* BufferLength */      sizeof requestReturn
                , /* StrLen_ */         &len );
rc = SQLFetch( hstmt );
if ( requestReturn[0] ) {
...
}
```

Notes

When called within a standalone transaction, ttRepSyncGet always returns the default value for requestReturn.

ttRepSyncGet may be called at any point within a transaction in which it is used to request the BY REQUEST return service for that transaction.

If you call ttRepSyncGet in a transaction that does not, in fact, update any RETURN RECEIPT BY REQUEST or RETURN TWOSAFE BY REQUEST replication elements, the call has no external effect.

See also

ttRepDeactivate
ttRepTransmitSet
ttReplicationStatus

`ttRepPolicySet`
`ttRepStart`
`ttRepStop`
`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
`ttRepSyncSet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepSyncSet

Description

This procedure sets static attributes associated with the caller's use of the replication-based return service. This procedure operates with either the `RETURN RECEIPT` or `RETURN TWOSAFE` service.

Required privilege

This procedure requires no privilege.

Syntax

`ttRepSyncSet(`*`requestReturn, returnWait, localAction`*`)`

Parameters

`ttRepSyncSet` has these optional parameters:

Parameter	Type	Description
requestReturn	`BINARY(1)`	0x00 - Turn off the return service for the current transaction. 0x01 - Turn on return services for the current transaction. Committing the transaction resets this attribute to its default value of 0 ("off"). You can use this parameter to turn on or turn off return services only when the replication subscribers have been configured with `RETURN RECEIPT BY REQUEST` or `RETURN TWOSAFE BY REQUEST`.
returnWait	`TT_INTEGER`	Specifies the number of seconds to wait for return service acknowledgment. The default value is 10 seconds. A value of '0' means there is no wait time. This timeout value overrides the value set by the `RETURN WAIT TIME` attribute in the `CREATE REPLICATION` or `ALTER REPLICATION` statement. The timeout set by this parameter persists across transaction boundaries and applies to all return services independent of the `BY REQUEST` option.

Parameter	Type	Description
localAction	TT_INTEGER	Action to be performed in the event the subscriber is unable to acknowledge commit of the transaction within the timeout period specified by *returnWait*. This parameter can only be used for return twosafe transactions. Set to NULL when using the RETURN service. 1 -- NO ACTION -- When a COMMIT times out, it returns the application unblocked, leaving the transaction in the same state it was when the COMMIT began,. The application may only reissue the COMMIT. This is the default. 2 -- COMMIT. When the COMMIT times out, the transaction is committed locally. No more operations are possible on this transaction, and the replicated databases diverge.This attribute persists across transactions and for the life of the connection.

Result set

ttRepSyncSet has no result set.

Examples

To enable the return receipt service in the current transaction for all the replication elements configured with RETURN RECEIPT BY REQUEST or RETURN TWOSAFE BY REQUEST, use:

```
rc = SQLExecDirect ( hstmt
, (SQLCHAR *)"{CALL ttRepSyncSet( 0x01 )}"
                    , SQL_NTS )
```

Note

The call to enable the return receipt service must be part of the transaction (AutoCommit must be off).

See also

ttRepDeactivate
ttRepTransmitSet
ttReplicationStatus
ttRepPolicySet
ttRepStart
ttRepStop
ttRepSubscriberStateSet
ttRepSubscriberWait

`ttRepSyncGet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepSyncSubscriberStatus

Description

This procedure queries a subscriber database in a replication scheme configured with a return service and a `RETURN DISABLE` failure policy to determine whether return service blocking for the subscriber has been disabled by the failure policy.

The `ttRepSyncSubscriberStatus` procedure returns the failure status of the subscriber database with the specified name on the specified host. You can specify only the *`storeName`*. However, an error is generated if the replication scheme contains multiple subscribers with the same name on different hosts.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRepSyncSubscriberStatus('storeName', 'hostName')
```

Parameters

`ttRepSyncSubscriberStatus` has these parameters:

Parameter	Type	Description
`subscriber`	`TT_VARCHAR (200) NOT NULL`	The name of the subscribing database to be queried.
`hostName`	`TT_VARCHAR (200)`	The host name of one or more stores that are configured to receive updates from the executing store; if `NULL`, then receiving stores are identified by receiver alone. If both receiver and hostname are `NULL`, then all receiving stores are selected.

Result set

`ttRepSyncSubscriberStatus` returns:

Column	Type	Description
`disabled`	`TT_INTEGER`	Value is either: '1', if the return service has been disabled on the subscriber database. '0' if the return service is still enabled on the subscriber database.

Note

If the replication scheme specifies `DISABLE RETURN ALL`, then you must use `ttRepSyncSubscriberStatus` to query the status of each individual subscriber in the replication scheme.

ttRepTransmitGet

Description

This procedure returns the status of transmission of updates to subscribers for the current transaction. The corresponding `ttRepSyncSet` built-in procedure allows you to stop transmission of updates to subscribers for the length of a transaction.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepTransmitGet()
```

Parameters

`ttRepTransmitGet` has no parameters.

Result set

`ttRepTransmitGet` returns the result:

Column	Type	Description
transmit	`TT_INTEGER`	0 - Updates are not being transmitted to any subscribers for the remainder of the transaction on the connection. 1- Updates are being transmitted to subscribers on the connection. (default)

Example

To return the transmit status on the active database in an active standby pair, use:

```
CALL ttRepTransmitGet();
```

See also

`ttRepDeactivate`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStateSave`
`ttRepStateSet`
`ttRepStop`
`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
`ttRepTransmitSet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepTransmitSet

Description

This procedure stops subsequent updates on the connection it is executed in from being replicated to any subscriber.

This procedure should be used with care since it could easily lead to transactional inconsistency of remote stores if partial transactions are replicated. If updates are disallowed from getting replicated, the subscriber stores diverge from the master store.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRepTransmitSet(transmit)
```

Parameters

`ttRepTransmitSet` has the parameter:

Parameter	Type	Description
transmit	`TT_INTEGER NOT NULL`	When set to 1 updates are transmitted to subscribers on the connection after the built-in is executed to replicate. (default) When set to 0 updates are not transmitted to any subscribers for the remainder of the transaction in which this call was issued and on the connection that issued it.

Result set

`ttRepTransmitSet` returns no results.

Example

To activate the active database in an active standby pair, use:

```
CALL ttRepTransmitSet(1);
```

To deactivate the active database in an active standby pair, use:

```
CALL ttRepTransmitSet(0);
```

See also

`ttRepDeactivate`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStateSave`
`ttRepStateSet`
`ttRepStop`
`ttRepSubscriberStateSet`

`ttRepSubscriberWait`
`ttRepTransmitGet`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepXactStatus

Description

This procedure checks on the status of a `RETURN RECEIPT` or `RETURN TWOSAFE` replication transaction. Using the built-in procedure `ttRepXactTokenGet`, you can get the token of a `RETURN RECEIPT` or `RETURN TWOSAFE` transaction. This is then passed as an input parameter to this built-in procedure. Only a token received from `ttRepXactTokenGet` may be used. The procedure returns a list of rows each of which have three parameters, a subscriber name, the replication status with respect to the subscriber and an error string that is only returned if a `RETURN TWOSAFE` replication transaction began but did not complete commit processing.

> **Note:** The error parameter is only returned for `RETURN TWOSAFE` transactions.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRepXactStatus(token)
```

Parameters

`ttRepXactStatus` has the parameter:

Parameter	Type	Description
xactID	`VARBINARY (10000)`	If no parameter is specified, status is returned for one of the following:- If called in a transaction that has begun, but not completed, commit processing, it returns the status of the transaction.- If called at any other time, it returns status for the most recently committed transaction on the connection that was in `RETURN RECEIPT` or `RETURN TWOSAFE` mode.

Result set

`ttRepXactStatus` returns the result set:

Column	Type	Description
subscriberName	`TT_CHAR (61)`	The name of the database that subscribes to tables updated in the transaction. The name returns as: *store_name@host_name*.

Column	Type	Description
state	TT_CHAR (2)	The state of the transaction with respect to the subscribing database. The return values are one of the following: 'NS' -- Transaction not sent to the subscriber. 'RC' -- Transaction received by the subscriber agent 'CT' - Transaction applied at the subscriber store. (Does not convey whether the transaction ran into an error when being applied.) 'AP' -- Transaction has been durably applied on the subscriber.
errorString	TT_VARCHAR (2000)	Error string retuned by the subscriber agent describing the error it encountered when applying the twosafe transaction. If no error is encountered, this parameter is NULL. Non-Null values are only returned when this procedure is called inside a twosafe replication transaction that has begun, but has not yet completed, processing a commit.

See also

ttRepDeactivate
ttRepTransmitSet
ttReplicationStatus
ttRepPolicySet
ttRepStart
ttRepStop
ttRepSubscriberStateSet
ttRepSubscriberWait
ttRepSyncGet
ttRepSyncSet
ttRepXactTokenGet
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttRepXactTokenGet

Description

This procedure returns a token for `RETURN RECEIPT` or `RETURN TWOSAFE` replication transactions. Depending on the input parameter, type, it returns either:

- A token to the most recently committed `RETURN RECEIPT` transaction on the connection handle in which it is invoked.
- A token to the most recent transaction on the connection handle in which it is invoked that has begun commit processing on a transaction in `RETURN TWOSAFE` mode.

This procedure can be executed in any subsequent transaction or in the same transaction after commit processing has begun for a transaction in `RETURN TWOSAFE` replication.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRepXactTokenGet('type')
```

Parameters

`ttRepXactTokenGet` has these parameters:

Parameter	Type	Description
type	`TT_CHAR (2) NOT NULL`	The type of transaction desired: `'RR'` -- Return receipt. `'R2'` -- Return twosafe.

Result set

`ttRepXactTokenGet` returns the result set:

Column	Type	Description
token	`VARBINARY (10000)`	A varbinary token used to represent the transaction desired.

See also

`ttRepDeactivate`
`ttRepTransmitSet`
`ttReplicationStatus`
`ttRepPolicySet`
`ttRepStart`
`ttRepStop`
`ttRepSubscriberStateSet`
`ttRepSubscriberWait`
`ttRepSyncGet`

`ttRepSyncSet`
`ttRepXactStatus`
"ttRepDuplicateEx" in *Oracle TimesTen In-Memory Database C Developer's Guide*

ttSetUserColumnID

Description

This procedure explicitly sets the value for the user-specified column ID. Updates presented to the application by the Transaction Log API may contain information about the columns of a table. This column information contains a system-specified column number and a user-specified column identifier. The user-specified column ID has the value 0 until set explicitly by this call.

The system assigns an ID to each column during a `CREATE` or `ALTER TABLE` operation. Setting a user-assigned value for the column ID allows you to have a unique set of column numbers across the entire database or a specific column numbering system for a given table.

Required privilege

This procedure requires the `XLA` privilege.

Syntax

```
ttSetUserColumnID('tblName', 'colName', repID)
```

Parameters

`ttSetUserColumnID` has these parameters:

Parameter	Type	Description
tblName	`TT_CHAR(61) NOT NULL`	Table name. Using a synonym to specify a table name is not supported.
colName	`TT_CHAR(30) NOT NULL`	Column name.
repID	`TT_INTEGER NOT NULL`	Integer identifier.

Result set

`ttSetUserColumnID` returns no results.

Example

```
CALL ttSetUserColumnID('APP.SESSION', 'SESSIONID', 15);
```

See also

`ttSetUserTableID`
Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide

ttSetUserTableID

Description

This procedure explicitly sets the value of the user table ID. The table that each row is associated with is expressed with two codes: an application-supplied code called the user table ID and a system-provided code called the system table ID. Updates are presented to the application by the Transaction Log API in the form of complete rows. The user table ID has the value zero until explicitly set with the `ttSetUserTableID` procedure.

Required privilege

This procedure requires the `XLA` privilege.

Syntax

```
ttSetUserTableID('tblName', repID)
```

Parameters

`ttSetUserTableID` has these parameters:

Parameter	Type	Description
tblName	`TT_CHAR (61) NOT NULL`	Table name. Using a synonym to specify a table name is not supported.
repID	BINARY(8) NOT NULL	Integer identifier.

Result set

`ttSetUserTableID` returns no results.

Example

```
CALL ttSetUserTableID('APP.SESSION', 0x123456);
```

See also

`ttSetUserColumnID`
Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide

ttSize

Description

This procedure estimates the size of a table or view. It returns a single row with a single DOUBLE column with the estimated number of bytes for the table. The table can be specified as either a table name or a fully qualified table name. A non-NULL *nrows* parameter causes the table size to be estimated assuming the statistics of the current table scaled up to the specified number of rows. If the *nrows* parameter is NULL, the size of the table is estimated with the current number of rows.

The current contents of the table are scanned to determine the average size of each VARBINARY and VARCHAR column. If the table is empty, the average size of each VARBINARY and VARCHAR column is estimated to be one-half its declared maximum size. The estimates computed by ttSize include storage for the table itself, VARBINARY and VARCHAR columns and all declared indexes on the table.

The table is scanned when this built-in procedure is called. The scan of the table can be avoided by specifying a non-NULL *frac* value, which should be between 0 and 1. This value is used to estimate the average size of varying-length columns. The maximum size of each varying-length column is multiplied by the *frac* value to compute the estimated average size of VARBINARY or VARCHAR columns. If the *frac* parameter is not given, the existing rows in the table are scanned and the average length of the varying-length columns in the existing rows is used. If *frac* is omitted and the table has no rows in it, then *frac* is assumed to have the value 0.5.

Required privilege

This procedure requires the SELECT privilege on the specified table.

Syntax

```
ttSize('tblName', nRows, frac)
```

Parameters

ttSize has these parameters:

Parameter	Type	Description
tblName	TT_CHAR(61) NOT NULL	Name of an application table. Can include table owner. This parameter is required. Using a synonym to specify a table name is not supported.
nRows	TT_INTEGER	Number of rows to estimate in a table. This parameter is optional.
frac	BINARY_DOUBLE	Estimated average fraction of VARBINARY or VARCHAR column sizes. This parameter is optional.

Result set

ttSize returns the result set:

Column	Type	Description
size	BINARY_DOUBLE NOT NULL	Estimated size of the table, in bytes.

Examples

```
CALL ttSize('ACCTS', 1000000, NULL);

CALL ttSize('ACCTS', 30000, 0.8);

CALL ttSize('SALES.FORECAST', NULL, NULL);
```

When using `ttSize`, you must first execute the command and then fetch the results. For example:

ODBC

```
double size;
SQLLEN len;

rc = SQLExecDirect(hstmt, "call ttSize('SalesData', 250000,
0.75)", SQL_NTS);
rc = SQLBindColumn(hstmt, 1, SQL_C_DOUBLE, &size, sizeof double,
&len);
rc = SQLFetch(hstmt);
rc = SQLFreeStmt(hstmt, SQL_CLOSE);
```

JDBC

```
. . . . . .
String URL="jdbc:timesten:MyDataStore";
Connection con;
double tblSize=0;
. . . . . .
con = DriverManager.getConnection(URL);
CallableStatement cStmt = con.prepareCall("
{CALL ttSize('SalesData', 250000, 0.75) }");
if( cStmt.execute() )
  {
   rs=cStmt.getResultSet();
   if (rs.next()) {
    tblSize=rs.getDouble(1);
   }
   rs.close();
  }
cStmt.close();
con.close();

. . . . . .
```

Note

The `ttSize` procedure allows you to estimate how large a table will be with its full population of rows based on a small sample. For the best results, we recommend populating the table with at least 1,000 typical rows.

ttSQLCmdCacheInfo

Description

This procedure returns all compiled SQL statements in the TimesTen SQL command cache.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttSQLCmdCacheInfo()
```

Parameters

`ttSQLCmdCacheInfo` has the optional parameter:

Parameter	Type	Description
sqlCommandID	`TT_INTEGER` for 32-bit systems `TT_BIGINT` for 64-bit systems	The unique identifier of a SQL command in the TimesTen command cache. If no value is supplied displays information about all current commands in the TimesTen cache.

Result set

`ttSQLCmdCacheInfo` returns the result set:

Parameter	Type	Description
sqlCommandID	`TT_INTEGER NOT NULL` for 32-bit systems `TT_BIGINT NOT NULL` for 64-bit systems	The unique identifier of a command in the TimesTen command cache.
privateCommandConnectionID	`TT_INTEGER`	The unique ID of a private connection. If not a private connection, the value is NULL.
executions	`TT_BIGINT` `NOT NULL`	A counter for the number of executions that took place on this command since it was brought into the command cache.
prepares	`TT_BIGINT` `NOT NULL`	A counter for the number of user prepares that result in a hit on the command cache.
reprepares	`TT_BIGINT` `NOT NULL`	A counter for the number of reprepares or invalidations of this command.

Parameter	Type	Description
freeable	TT_TINYINT NOT NULL	Indicates whether this command can be garbage collected by the subdaemon. **1** - Indicates freeable. **0** - Indicates non-freeable.
size	TT_INTEGER NOT NULL	The total space (bytes) allocated for this command in the command cache.
owner	TT_CHAR(31) NOT NULL	The identifier of the user who created this command.
queryText	TT_VARCHAR(409600) NOT NULL	The full SQL text for the current command.

Examples

To display command info for all the current valid commands, use

```
Command> CALL ttSQLCmdCacheInfo;
< 528079360, 2048, 0, 1, 0, 1, 2168, TTUSER , select * from t7 where x7 is
not null or exists (select 1 from t2,t3 where not 'tuf' like 'abc') >
< 527609108, 2048, 0, 1, 0, 1, 2960, TTUSER , select * from t1 where x1 =
(select x2 from t2 where z2 in (1,3) and y1=y2) order by 1, 2, 3 >
< 528054656, 2048, 0, 1, 0, 1, 1216, TTUSER , create table t2(x2 int,y2 int,
z2 int) >
< 528066648, 2048, 0, 1, 0, 1, 1176, TTUSER , insert into t2 select * from t1 >
< 528013192, 2048, 0, 1, 0, 1, 1848, TTUSER , select * from t1 where exists
 (select * from t2 where x1=x2) or y1=1 >
< 527582620, 2048, 0, 1, 0, 1, 1240, TTUSER , insert into t2 select * from t1 >
< 527614292, 2048, 0, 1, 0, 1, 2248, TTUSER , select * from t1 where exists
 (select x2 from t2 where x1=x2) order by 1, 2, 3 >
< 528061248, 2048, 0, 1, 0, 1, 696, TTUSER , create index i1 on t3(y3) >
< 528070368, 2048, 0, 1, 0, 1, 824, TTUSER , call ttOptSetOrder('t3 t4 t2 t1') >
< 528018856, 2048, 0, 1, 0, 1, 984, TTUSER , insert into t2 select * from t1 >
< 527606460, 2048, 0, 1, 0, 1, 2624, TTUSER , select * from t1 where x1 =
(select x2 from t2 where y1=y2) order by 1, 2, 3 >
< 528123000, 2048, 0, 1, 0, 1, 3616, TTUSER , select * from t1 where x1 = 1 or
x1 = (select x2 from t2,t3 where z2=t3.x3) >
< 528074624, 2048, 0, 1, 0, 1, 856, TTUSER , call ttOptSetOrder('t4 t2 t3 t1') >
< 527973892, 2048, 0, 1, 0, 1, 2872, TTUSER , select * from t1 where x1 in
(select x2 from t2) or x1 in (select x3 from t3) order by 1, 2, 3 >
< 527953876, 2048, 0, 1, 0, 1, 3000, TTUSER , select * from t1 where x1 =
(select x2 from t2) order by 1, 2, 3 >
< 527603900, 2048, 0, 1, 0, 1, 2440, TTUSER , select * from t1 where x1 in
(select x2 from t2 where y1=y2) order by 1, 2, 3 >
< 528093308, 2048, 0, 1, 0, 1, 3608, TTUSER , select * from t1 where x1 = 1 or
x1 = (select x2 from t2,t3 where z2=t3.x3 and t3.z3=1) >
< 528060608, 2048, 0, 1, 0, 1, 696, TTUSER , create index i1 on t2 (y2) >
…..
```

To display command info for the SqlCmdId 527973892, use

```
Command> CALL ttSQLCmdCacheInfo(527973892);
< 527973892, 2048, 0, 1, 0, 1, 2872, TTUSER,
select * from t1 where x1 in (select x2 from t2) or
x1 in (select x3 from t3) order by 1, 2, 3 >
1 row found.
```

ttSQLCmdCacheInfoGet

Description

This procedure returns command statistics update information. This procedure can be useful for certain operations, such as `CREATE INDEX` or `DROP INDEX` and other DDL operations that alter objects.

Required privilege

This procedure requires no privilege.

Syntax

```
ttSQLCmdCacheInfoGet()
```

Parameters

`ttSQLCmdCacheInfoGet` has no parameters.

Result set

`ttSQLCmdCacheInfoGet` returns the result set:

Parameter	Type	Description
cmdCount	`TT_INTEGER NOT NULL`	Number of commands in the cache
freeableCount	`TT_INTEGER NOT NULL`	Count of number of freeable commands that can be garbage collected by the subdaemon at that moment. This number is obtained by examining the command information.
size	`TT_BIGINT NOT NULL`	The current total space allocated to store all the cached commands.

Examples

To generate a list of all TimesTen instance users, use:

```
Command> CALL ttSQLCmdCacheInfoGet;
< 5,4,12316 >
1 row found
```

ttSQLCmdQueryPlan

Description

This procedure returns all detailed run-time query plans for SQL statements in the TimesTen SQL command cache. If no argument is supplied, this procedure displays the query plan for all valid commands in the TimesTen cache. For invalid commands, an error is returned that displays the text of the query and the syntax problems.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttSQLCmdQueryPlan(commandID)
```

Parameters

`ttSQLCmdQueryPlan` has the optional parameter:

Parameter	Type	Description
sqlCommandID	`TT_INTEGER` for 32-bit systems `TT_BIGINT` for 64-bit systems	The unique identifier of a SQL command in the TimesTen command cache. If no value is supplied displays the query plan for all valid commands in the TimesTen cache.

Result set

`ttSQLCmdQueryPlan` returns the result set:

Parameter	Type	Description
sqlCommandID	`TT_INTEGER NOT NULL` for 32-bit systems `TT_BIGINT NOT NULL` for 64-bit systems	The unique identifier of a command in the TimesTen command cache.
queryText	`TT_VARCHAR(1024)`	The first 1024 characters of the SQL text for the current command.
step	`TT_INTEGER`	The step number of current operation in this run-time query plan.
level	`TT_INTEGER`	The level number of current operation in this run-time query plan.
operation	`TT_CHAR(31)`	The operation name of the current step in this run-time query plan.

Parameter	Type	Description
tblName	TT_CHAR(31)	Name of the table used in this step, if any. Using a synonym to specify a table name is not supported.
tblOwnerName	TT_CHAR(31)	Name of the owner of the table used in this step, if any.
ixName	TT_CHAR(31)	Name of the index used in this step, if any.
indexedPred	TTVARCHAR(1024)	In this step, if an index is used, the indexed predicate is printed if available. Not all expressions can be printed out and the output may be fragmented and truncated. "..." represents the unfinished portion of the expression.
nonIndexedPred	TT_VARCHAR(1024)	In this step, if a non-indexed predicate is used, the non-indexed predicate is printed if available. Not all expressions can be printed out and the output may be fragmented and truncated. "..." represents the unfinished portion of the expression.

Examples

To display the query plan for `SQLCmdID` `528078576`:

```
Command> call ttSqlCmdQueryPlan(528078576);
< 528078576, select * from t1 where 1=2 or (x1 in (select x2 from t2, t5 where
y2 in (select y3 from t3)) and y1 in (select x4 from t4)), <NULL>, <NULL>, <NULL>,
 <NULL>, <NULL>, <NULL>, <NULL>, <NULL> >
< 528078576, <NULL>, 0, 4, RowLkSerialScan , T1 , TTUSER , , , >
< 528078576, <NULL>, 1, 7, RowLkTtreeScan , T2 , TTUSER , I2 , , >
< 528078576, <NULL>, 2, 7, RowLkTtreeScan , T5 , TTUSER , I2 , , >
< 528078576, <NULL>, 3, 6, NestedLoop , , , , , >
< 528078576, <NULL>, 4, 6, RowLkTtreeScan , T3 , TTUSER , I1 , ( (Y3=Y2; ) ) , >
< 528078576, <NULL>, 5, 5, NestedLoop , , , , , >
< 528078576, <NULL>, 6, 4, Filter , , , , , X1 = X2; >
< 528078576, <NULL>, 7, 3, NestedLoop(Left OuterJoin) , , , , , >
< 528078576, <NULL>, 8, 2, Filter , , , , , >
< 528078576, <NULL>, 9, 2, RowLkTtreeScan , T4 , TTUSER , I2 , , Y1 = X4; >
< 528078576, <NULL>, 10, 1, NestedLoop(Left OuterJoin) , , , , , >
< 528078576, <NULL>, 11, 0, Filter , , , , , >
13 rows found.
```

To display query plans for all valid queries, omit the argument for `ttSqlCmdQueryPlan`:

```
< 528079360, select * from t7 where x7 is not null or exists (select 1 from t2,t3
 where not 'tuf' like 'abc'), <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>,
 <NULL>, <NULL> >
< 528079360, <NULL>, 1, 3, RowLkTtreeScan  , T2
 , TTUSER  , I2 , , NOT(LIKE( tuf ,abc ,NULL )) >
< 528079360, <NULL>, 2, 3, RowLkTtreeScan  , T3  , TTUSER  , I2  , , >
< 528079360, <NULL>, 3, 2, NestedLoop  ,  ,  ,  , , >
```

```
< 528079360, <NULL>, 4, 1, NestedLoop(Left OuterJoin) ,    ,    ,    , , >
< 528079360, <NULL>, 5, 0, Filter    ,    ,    ,    , , X7 >
< 527576540, call ttSqlCmdQueryPlan(527973892), <NULL>, <NULL>, <NULL>, <NULL>,
 <NULL>, <NULL>, <NULL>, <NULL> >
< 527576540, <NULL>, 0, 0, Procedure Call  ,  ,  ,  , , >
< 528054656, create table t2(x2 int,y2 int, z2 int), <NULL>, <NULL>, <NULL>,
<NULL>, <NULL>, <NULL>, <NULL>, <NULL> >
< 528066648, insert into t2 select * from t1, <NULL>, <NULL>, <NULL>, <NULL>,
<NULL>, <NULL>, <NULL>, <NULL> >
< 528066648, <NULL>, 0, 0, Insert  , T2  , TTUSER  ,  , , >
< 528013192, select * from t1 where exists (select * from t2 where x1=x2) or y1=1,
<NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL> >
< 528061248, create index i1 on t3(y3), <NULL>, <NULL>, <NULL>, <NULL>, <NULL>,
<NULL>, <NULL>, <NULL> >
< 528070368, call ttOptSetOrder('t3 t4 t2 t1'), <NULL>, <NULL>, <NULL>, <NULL>,
<NULL>, <NULL>, <NULL>, <NULL> >
< 528070368, <NULL>, 0, 0, Procedure Call  ,  ,  ,  , , >
< 528018856, insert into t2 select * from t1, <NULL>, <NULL>, <NULL>, <NULL>,
<NULL>, <NULL>, <NULL>, <NULL> >
< 527573452, call ttsqlCmdCacheInfo(527973892), <NULL>, <NULL>, <NULL>, <NULL>,
<NULL>, <NULL>, <NULL>, <NULL> >
< 527573452, <NULL>, 0, 0, Procedure Call   ,    ,    ,    , , >
….. /* more rows here */
```

ttVersion

Description

This procedure returns the 5 digits that make up the release number of the current TimesTen instance.

Required privilege

This procedure requires no privilege.

Syntax

```
ttVersion()
```

Parameters

`ttVersion` has no parameters:

Result set

`ttVersion` returns the result set:

Column	Type	Description
major	`TT_INTEGER NOT NULL`	The major release number. Indicates releases with major infrastructure and functionality changes.
minor	`TT_INTEGER NOT NULL`	The minor release number. Indicates a version with new functionality changes, but no infrastructure changes.
patch	`TT_INTEGER NOT NULL`	The patch release number. Indicates a release that contains all bug fixes since the previous maintenance release.
relDot4	`TT_INTEGER NOT NULL`	Indicates a release with minor bug fixes.
relDot5	`TT_INTEGER NOT NULL`	This digit in the release number is reserved for future use.

Example

```
CALL ttVersion( );
<11, 2, 1 , 1, 0>
1 row found.
```

In this case, the TimesTen release number is: 11.2.1.1.0.

ttWarnOnLowMemory

Description

This procedure allows applications to specify that operations executed on the current connection should return a warning if they allocate memory and find that memory is low. If the value is set, a warning is returned for any operation that does an allocation and finds total memory in use to be above the connection's threshold value as specified by the `PermWarnThreshold` and `TempWarnThreshold` database attributes. See "Connection Attributes" on page 1-1 for more information.

Required privilege

This procedure requires no privilege.

Syntax

```
ttWarnOnLowMemory(permanent, temporary)
```

Parameters

`ttWarnOnLowMemory` has these parameters:

Parameter	Type	Description
permanent	`TT_INTEGER NOT NULL`	1(enable) or 0 (disable) warnings for the permanent data partition.
temporary	`TT_INTEGER NOT NULL`	1 (enable) or 0 (disable) warnings for the temporary data partition.

Result set

`ttWarnOnLowMemory` returns no results.

Example

```
CALL ttWarnOnLowMemory(1, 0);
```

Enables low memory warnings for the permanent data partition only.

Note

By default, low memory warnings are not issued for either partition. Applications that want to receive these warnings must call this procedure. This procedure is connection specific, and so you must issue it for each connection upon which warnings are desired. Also, the current setting does not persist to subsequent connections.

ttXactIdGet

Description

This procedure returns transaction ID information for interpreting lock messages. The two result columns of `ttXactIdGet` are used in combination to uniquely identify a transaction in a database. Taken individually, the columns are not interesting. The result should only be used to correlate with other sources of transaction information. The numbers may not follow a strict pattern.

Required privilege

This procedure requires no privilege.

Syntax

```
ttXactIdGet()
```

Parameters

`ttXactIdGet` has no parameters.

Result set

`ttXactIdGet` returns the result set:

Column	Type	Description
xactID	TT_INTEGER	Connection ID.
counter	TT_BIGINT	An increasing number that is used to discriminate successive transactions of the same Transaction Id.

Example

```
Command > automcommit 0;
Command > call ttXactIdGet;
<2,11>
1 row found
Command > commit;
Command > call ttXactIdGet
<3, 12>
1 row found
```

Note

The output correlates to the values printed in lock error messages and `ttXactAdmin` lock information output.

See also

`ttXactAdmin`
"ttXactIdRollback" in the *Oracle TimesTen In-Memory Database C Developer's Guide.*

ttXlaBookmarkCreate

Description

This procedure creates the specified bookmark.

Required privilege

This procedure requires the `XLA` privilege.

Syntax

```
ttXlaBookmarkCreate('bookmark', 'replicated')
```

Parameters

`ttXlaBookmarkCreate` has the parameter:

Parameter	Type	Description
Parameter	Type	Description
bookmark	`TT_CHAR (31) NOT NULL`	The name of the bookmark to be created.
replicated	BINARY(1)	0x00 or null (equivalent) for non-replicated bookmarks (default setting); 0x01 for replicated bookmarks. If null, non-replicated bookmarks are used.

Result set

`ttXlaBookmarkCreate` returns no results.

Example

For non-replicated bookmark, execute the following:

```
Command > call ttXlaBookmarkCreate(mybookmark);
```

or

```
Command> call ttxlabookmarkcreate('mybkmk2',0x00);
```

For a replicated bookmark, execute the following:

```
Command > call ttXlaBookmarkCreate(mybookmark, 0x01);
```

For more details on XLA bookmarks, including replicated XLA bookmarks, see "About XLA bookmarks" in the *Oracle TimesTen In-Memory Database C Developer's Guide*.

See also

`ttXlaSubscribe`
`ttXlaUnsubscribe`
`ttXlaBookmarkDelete`

ttXlaBookmarkDelete

Description

This procedure deletes the specified bookmark. The bookmark cannot be deleted while it is in use.

Required privilege

This procedure requires the `XLA` privilege.

Syntax

```
ttXlaBookmarkDelete('bookmark')
```

Parameters

`ttXlaBookmarkDelete` has the parameter:

Parameter	Type	Description
bookmark	TT_CHAR (31) NOT NULL	The name of the bookmark to be deleted.

Result set

`ttXlaBookmarkDelete` returns no results.

Example

```
Command > call ttXlaBookmarkDelete('mybookmark');
```

Note

Before dropping a table that is subscribed to by an XLA bookmark, you must first drop all XLA bookmarks or unsubscribe from XLA tracking.

See also

`ttXlaBookmarkCreate`
`ttXlaSubscribe`
`ttXlaUnsubscribe`

ttXlaSubscribe

Description

This procedure configures persistent XLA tracking of a table. This procedure cannot be executed when the specified bookmark is in use.

Required privilege

This procedure requires the `XLA` privilege.

Syntax

```
ttXlaSubscribe('tblName', 'bookmark')
```

Parameters

`ttXlaSubscribe` has the parameters:

Parameter	Type	Description
tblName	TT_CHAR (61) NOT NULL	The name of the table to be tracked. Using a synonym to specify a table name is not supported.
bookmark	TT_CHAR (31) NOT NULL	The name of the bookmark that the application uses to track this table.

Result set

`ttXlaSubscribe` returns no results.

Example

```
Command > call ttXlaSubscribe ('SALLY.ACCTS', mybookmark);
```

See also

`ttXlaBookmarkCreate`
`ttXlaBookmarkDelete`
`ttXlaUnsubscribe`

ttXlaUnsubscribe

Description

This procedure stops persistent XLA tracking of a table. This procedure cannot be executed when the specified bookmark is in use.

Required privilege

This procedure requires the `XLA` privilege.

Syntax

```
ttXlaUnsubscribe('tblName', 'bookmark')
```

Parameters

`ttXlaUnsubscribe` has the parameters:

Parameter	Type	Description
tblName	TT_CHAR (61) NOT NULL	The name of the table on which XLA tracking should be stopped. Using a synonym to specify a table name is not supported.
bookmark	TT_CHAR (31) NOT NULL	The name of the bookmark that the application uses to track this table.

Result set

`ttXlaSubscribe` returns no results.

Example

```
Command > call ttXlaSubscribe ('SALLY.ACCTS', mybookmark);
```

Note

Before dropping a table that is subscribed to by an XLA bookmark, you must first drop all XLA bookmarks or unsubscribe from XLA tracking.

See also

`ttXlaBookmarkCreate`
`ttXlaBookmarkDelete`
`ttXlaSubscribe`

3

Utilities

The following sections provide reference information and examples for TimesTen utilities.

- Overview
- Required authentication and authorization for utilities

Overview

The options for TimesTen utilities are generally not case sensitive, except for single character options. You can use `-connstr` or `-connStr` interchangeably. However `-v` and `-V` are each unique options.

All utilities return `0` for success and non-zero if an error occurs.

Note: The utility name and options listed in this chapter are case-insensitive. They are described in mixed case to make the examples and syntax descriptions easier to read.

Required authentication and authorization for utilities

The following sections describe the authentication and authorization required for utilities:

- Required user authentication for utilities
- Required privileges for executing utilities

Required user authentication for utilities

All utilities that require a password prompt for one.

If a `UID` attribute is given but no `PWD` attribute is given, either through a connection string or in the `ODBCINI` file for the specified DSN, TimesTen prompts for a password. When explicitly prompted, input is not displayed on the command line.

A password given on the command line, before TimesTen prompts for the password, is visible to the `ps` command, so use of the `PWD` attribute is not recommended in the first call to the utility. For example, the following command is not recommended:

```
% ttIsql -connStr "DSN=mydsn;UID=terry;PWD=secret"
```

Generally, when no `UID` attribute is given, the `UID` is assumed to be the user name identified by the operating system, and TimesTen does not prompt for a password.

When a utility accepts a DSN, connection string or database path as a parameter, specify the value at the end of the command line.

Required privileges for executing utilities

Certain TimesTen command-line utilities require privileges. Each utility in this chapter describes the privilege required for execution. You may receive a "database not loaded" error if you try to execute any utility with a user other than the instance administrator and the database is not loaded into memory. In this case, we cannot determine the privileges of the user. Thus any utilities requiring privileges have to be run either as the instance administrator or executed while the database is loaded.

ttAdmin

Description

Allows you to:

- Specify policies to automatically or manually load and unload databases from RAM.
- Specify policies to automatically or manually start and stop replication agents for specified databases.
- Start and stop TimesTen cache agents for caching data from Oracle tables. The cache agent is a process that handles Oracle database access on behalf of a TimesTen database. It also handles the aging and autorefresh of the cache groups in the TimesTen database. Before using any cache features, you must start the cache agent. Cache options require that you specify a value for the OracleNetServiceName in the DSN.

Required privilege

This utility requires no privileges to query the database.

Replication options require the ADMIN privilege.

Cache options require the CACHE_MANAGER privilege.

All other options require the ADMIN privilege.

If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttAdmin {-h | -help | -?}

ttAdmin {-V | -version}

ttAdmin [-ramPolicy always|manual|inUse [-ramGrace secs] ]
[-ramLoad] [-ramUnload]
[-repPolicy always|manual|norestart]
[-reqpQueryThresholdGet]
[-reqpQueryThresholdSet seconds]
[-repStart | -repStop]
[[-cacheUidGet] |
 [-cacheUidPwdSet -cacheUid uid [-cachePwd pwd]] |
 [-cachePolicy always | manual| norestart] |
 [-cacheStart] |
 [-cacheStop [-stopTimeout seconds]]]
[-query]
{-connStr connection_string | DSN}
```

Options

`ttAdmin` has the options:

Option	Description
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the database to be administered.
`-h -help` `-?`	Prints a usage message and exits.
`-cachePolicy`	Defines the policy used to determine when the cache agent for the database should run. This option requires CACHE_MANAGER privileges.
`-cacheStart`	Starts a cache agent for the database. This option requires CACHE_MANAGER privileges.
`-cacheStop`	Stops a cache agent for the database. You should not shut down the cache agent immediately after dropping or altering a cache group. Instead, wait for at least two minutes. Otherwise, the cache agent may not get a chance to clean up the Oracle objects that were used by the AUTOREFRESH feature. This option requires CACHE_MANAGER privileges.
`-cachePwd`	The password associated with the cache administration user ID that is used to manage AUTOREFRESH cache groups and ASYNCHRONOUS WRITETHROUGH cache groups. The cache administration user has extended privileges. See "Grant privileges to the Oracle users" in the *Oracle In-Memory Database Cache User's Guide* for more details. This option requires CACHE_MANAGER privileges.
`-cacheUid`	The cache administration user ID. The cache administration user manages AUTOREFRESH cache groups and ASYNCHRONOUS WRITETHROUGH cache groups. The cache administration user has extended privileges. This option requires CACHE_MANAGER privileges. See "Grant privileges to the Oracle users" in the *Oracle In-Memory Database Cache User's Guide* for more details.
`-cacheUidGet`	Gets the current cache administration user ID for the specified database. This option requires CACHE_MANAGER privileges.
`-cacheUidPwdSet`	Sets the cache administration user ID and password for the specified database. This option requires CACHE_MANAGER privileges. Must be set with the `-cacheUid` and `-cachePwd` options. Some things to consider are: ▪ You only need to specify the cache administration user ID and password once for each new database. ▪ You cannot set the cache administration user ID and password while either the cache agent or the replication agent are running. ▪ The cache administration user ID cannot be reset while there are ASYNCHRONOUS WRITETHROUGH cache groups or AUTOREFRESH cache groups (with a state that is not equal to OFF) on the database. ▪ The cache administration password can be changed at any time.

Option	Description
`-query`	Displays a summary of the policy settings for the named database.
`-ramGrace` *`secs`*	Only effective if `-ramPolicy` is `inUse`. If nonzero, the database is kept in RAM for secs seconds before being unloaded after the last application disconnects from the database.
`-ramLoad`	Valid only when `-ramPolicy` is set to `manual`. Causes the database to be loaded into RAM.
`-ramPolicy` *`policy`*	Defines the policy used to determine when the database is loaded into system RAM. ■ `always` - Specifies that the database should remain in system RAM all the time. ■ `manual`- Specifies that the database is only to be loaded in system RAM when explicitly loaded by the user (using the ramLoad option). ■ `inUse` - (default) Specifies that the database is only loaded in system RAM when in use (i.e., when applications are connected). The `-ramGrace` option may be used to modify the behavior of this policy.This option cannot be used with temporary databases. TimesTen only allows a temporary database to be loaded into RAM manually. Trying to set the policy generates a warning. This option requires ADMIN privileges.
`-ramUnload`	Valid only when `-ramPolicy` is set to `manual`. Causes the database to be unloaded from RAM.
`-repPolicy`	Defines the policy used to determine when the replication agent starts. ■ `always` - Specifies that the agent should always be running for the database. This option immediately starts the replication agent and when the daemon restarts the replication agent is restarted. ■ `manual`- (default) Specifies that the replication agent must be manually started and stopped. ■ `norestart` - Specifies that the replication agent for the database is not to be restarted after a failure. This option requires ADMIN privileges.
`-repPolicy`	Defines the policy used to determine when the replication agent starts. ■ `always` - Specifies that the agent should always be running for the database. This option immediately starts the replication agent and when the daemon restarts the replication agent is restarted. ■ `manual`- (default) Specifies that the replication agent must be manually started and stopped. ■ `norestart` - Specifies that the replication agent for the database is not to be restarted after a failure. This option requires ADMIN privileges.

Option	Description
`-repQueryThresholdGet`	Returns the number of seconds that a query can be executed by the replication agent before TimesTen writes a warning to the support log and throws an SNMP trap. A value of 0 and indicates that no warning is sent. This option requires ADMIN privileges.
`-repQueryThresholdSet`	This option specifies the number of seconds that a query can be executed by the replication agent before TimesTen writes a warning to the support log and throws an SNMP trap. The specified value takes effect the next time the replication agent is started. The query threshold for the replication agent applies to SQL execution on detail tables of materialized views, ON DELETE CASCADE operations and some internal operations. .The value must be greater than or equal to 0. Default is 0 and indicates that no warning is sent. This option requires ADMIN privileges.
`-repStop`	Stops the database's replication agent.
`-stopTimeout` *`seconds`*	Specifies that the TimesTen daemon should stop the cache agent if it doesn't stop within *seconds*. If set to 0, the daemon waits forever for the cache agent. The default value is 100 seconds. This option requires CACHE_MANAGER privileges.
`-V` | `-version`	Prints the release number of `ttAdmin` and exits.

Examples

A database referred to by DSN `SalesData` is used by some very performance sensitive applications. So that applications do not have to wait for the database to be loaded from disk into RAM, this database must always remain in RAM. To achieve this, use:

```
ttAdmin -ramPolicy always SalesData
```

The SalesData database is normally always resident in RAM. However, it is not being used at all today and should be loaded only when applications are connected to it. To change the RAM policy, use:

```
ttAdmin -ramPolicy inUse SalesData
```

To manually control whether the SalesData database is loaded into RAM and to *`load`* it now, use:

```
ttAdmin -ramPolicy manual -ramLoad SalesData
```

To manually *`unload`* the SalesData database from RAM, thus preventing any new applications from connecting to the database, use:

```
ttAdmin -ramPolicy manual -ramUnload SalesData
```

A database referred to by DSN `History` is not always in use. Permanently loading it into RAM unnecessarily uses memory. This database is idle for long periods, but when it is in use multiple users connect to it in rapid succession. To improve performance, it may be best to keep the database in RAM when applications are connected to it and to

keep it in RAM for 5 minutes (300 seconds) after the last user disconnects. With this RAM policy, as long as applications are connected to the database, the database remains in RAM. To set this policy, use:

```
ttAdmin -ramPolicy inUse -ramGrace 300 History
```

A database referred to by DSN SalesData is used to cache Oracle data. Use the following `ttAdmin` command to start the cache agent for the SalesData DSN:

```
ttAdmin -cacheStart SalesData
```

You can also use the `-cachePolicy` option to ask the TimesTen data manager daemon to start the cache agent every time the data manager itself is started. Use:

```
ttAdmin -cachePolicy always SalesData
```

To turn off the automatic start of cache agent, use:

```
ttAdmin -cachePolicy manual SalesData
```

To set the cache administration user ID and password, the `-cacheUidPwdSet` flag with the `-cacheUid` and `-cachePwd` options can be used with `ttAdmin`. For example, if the cache administration user ID and password on the database `SalesData` should be `scott` and `tiger` respectively, use:

```
ttAdmin -cacheUidPwdSet -cacheUid scott -cachPwd tiger SalesData
```

To get the current cache administration user ID for the SalesData DSN, use:

```
ttAdmin -cacheUidGet SalesData
```

ttAdmin displays the following output:

```
Cache User Id: scott
RAM Residence Policy: inUse
Replication Agent Policy: manual
Replication Manually Started: False
Cache Agent Policy: manual
Cache Agent Manually Started: False
```

Notes

If TimesTen is installed as a user instance, and the user attempts to start the cache agent for a database with a relative path, TimesTen looks for the database relative to where it is running, and fails. Therefore, a relative path should not be used in this scenario. For example, on Windows, if you have specified the path for the database as `DataStore=./dsn1` and attempt to start the cache agent with the command `ttAdmin -cacheStart dsn1`, the cache agent does not start because it looks for the database in *`install_dir`*`\srv\dsn1`. For UNIX it looks in a directory in `/var/TimesTen/`*`instance`*`/`.

When using AUTOREFRESH (automatic propagation from Oracle to TimesTen) or ASYNCHRONOUS WRITETHROUGH cache groups, you must specify the cache administration user ID and password. This user account is used to perform AUTOREFRESH and ASYNCHRONOUS WRITETHROUGH operations.

To load Oracle data, the TimesTen cache agent must be running. This requires that the ORACLE_HOME environment variable be set to the path of the Oracle installation. See the *Oracle In-Memory Database Cache User's Guide* for more details. For details on other environment variables that you may want to set, see "Environment variables" in the *Oracle TimesTen In-Memory Database Installation Guide*.

This utility is supported only for TimesTen Data Manager DSNs. It is not supported for TimesTen Client DSNs.

If `ttAdmin` is used with `-repStart` and it does not find a replication definition, the replication agent is not started and `ttAdmin` prints out an error message. For example:

```
$ ttAdmin -repstart repl1
*** [TimesTen][TimesTen 11.2.1.0.0 ODBC Driver][TimesTen]TT8191:
This store (repl1 on my_host) is not involved in a replication scheme --
file "eeProc.c", lineno 11016, procedure "RepAdmin()"
*** ODBC Error = S1000, TimesTen Error = 8191
```

If `ttAdmin` is used with the `-ramPolicy always` option, a persistent system connection is created on the database. The `-ramPolicy always` option can only be specified on shared databases.

The only `-ramPolicy` value supported for temporary databases is the `-ramPolicy manual` option with the `-ramLoad` option specified at the same time.

If `ttAdmin` is used with `-repPolicy manual` (the default) or `-repPolicy always`, then the `-ramPolicy always` option should also be used. This ensures that the replication agent begins recovery after a failure as quickly as possible.

See also

ttStatus
ttCachePolicySet
ttCacheUidGet
ttCacheUidPwdSet
ttCacheStart
ttCacheStop

ttAdoptStores

Description

On UNIX systems, moves databases from a TimesTen instance to a new TimesTen instance that is of the same minor version. For example, you can move files from TimesTen 11.2.1.0.0 to TimesTen 11.2.1.0.1.

The utility makes it easy to move to a new patch release and can be useful for testing a new release without uninstalling the old one. You can install the new version of TimesTen and move one or more databases to the new release to test it without uninstalling the old one. Both utilities require that both instances be of the same minor version.

You must run the `ttAdoptStores` utility from the destination instance.

Required privilege

This utility must be run by the TimesTen instance administrator. The instance administrator must be the same user for both the old and new TimesTen instance.

Syntax

```
ttadoptstores {-h | -help | -?}
ttadoptstores {-V | -version}
ttadoptstores [-quiet] -dspath path
ttadoptstores [-quiet] -instpath path
```

Options

`ttAdoptStores` has the options:

Option	Description
`-dspath` *`path`*	Adopts a single database. The path argument must be the path to the database files (without any file extensions).

`-h`
`-help`
`?`
	Prints a usage message and exits.

`-instpath` *`path`*	Adopts all databases for an instance. The path argument must be the path to the daemon working directory ("`infodir`"). If any databases are in use, the utility fails without making any modifications. No new connections to any database are allowed in the source instance until the entire operation has completed.
`-quiet`	Do not return verbose messages.
`-V | -version`	Prints the release number of `ttAdoptStores` and exits.

Examples

To adopt the database `/my/data/stores/ds`, use:

```
ttadoptstores -dspath /my/data/stores/ds
```

To adopt all the databases in the directory `/opt/TimesTen/ instance1`, use:

```
ttadoptstores -instpath /opt/TimesTen/instance1
```

Note

You cannot adopt temporary databases.

If an instance being adopted is part of a replication scheme, port numbers must match on each side of the replication scheme, unless a port number was specified as the value of the `-remoteDaemonPort` option during a `ttRepAdmin -duplicate` operation. Generally, all instances involved in the replication scheme must be updated at the same time.

This utility does not copy any `sys.odbc.ini` entries. You must move these files manually.

ttBackup

Description

Creates a backup copy of a database that can be restored at a later time using the `ttRestore` utility. If the database is in use at the time of backup, it must be in *shared* mode to enable `ttBackup`. For an overview of the TimesTen backup and restore facility, see "Migration, backup, and restoration of the database" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This utility requires the ADMIN privilege.

If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttBackup {-h | -help | -?}
ttBackup {-V | -version}
ttBackup -dir directory [-type backupType]
[-fname fileprefix] [-force]
{-connStr connection_string | DSN}
```

Options

`ttBackup` has the options:

Option	Description
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the database to be backed up.
`-dir` *`directory`*	Specifies the directory where the backup files should be stored.
`-fname` *`fileprefix`*	Specifies the file prefix for the backup files in the backup directory. The default value for this option is the file name portion of the `DataStore` parameter of the database's ODBC definition.
`-force`	Forces the backup into the specified directory. If a backup exists in that directory, `ttBackup` overwrites it. If this option is not specified `ttBackup` terminates with an end message without overwriting existing files.
`-h -help -?`	Prints a usage message and exits.

Option	Description
`-type` *`backupType`*	Specifies the type of backup to be performed. Valid values are: ▪ `fileFull` - Performs a full file backup to the backup path specified by the *`directory`* and *`fileprefix`* parameters (default). The resulting backup is not enabled for incremental backup. ▪ `fileFullEnable` - Performs a full file backup to the backup path specified by the *`directory`* and *`fileprefix`* parameters. The resulting backup is enabled for incremental backup. ▪ `fileIncremental` - Performs an incremental file backup to the backup path specified by the *`directory`* and *`fileprefix`* parameters, if that backup path contains an incremental-enabled backup of the database. Otherwise, an error is returned. ▪ `fileIncrOrFull` - Performs an incremental file backup to the backup path specified by the *`directory`* and *`fileprefix`* parameters if that backup path contains an incremental-enabled backup of the database. Otherwise, it performs a full file backup of the database and marks it incremental enabled. ▪ `streamFull`- Performs a stream backup to standard out ▪ `incrementalStop`- Does not perform a backup. Disables incremental backups for the backup path specified by the *`directory`* and *`fileprefix`* parameters. This prevents transaction log files from accumulating for an incremental backup.
`-V` | `-version`	Prints the release number of `ttBackup` and exits.

Examples

To perform a full file backup of the FastIns database to the `backup` directory in`/users/pat/TimesTen/backups`, use:

```
ttBackup -type fileFullEnable -dir /users/pat/TimesTen/backups FastIns
```

To copy the `FastIns` database to the file `FastIns.back`, use:

```
ttBackup -type streamFull FastIns > FastIns.back
```

On UNIX, to save the `FastIns` database to a backup tape, use:

```
ttBackup -type streamFull FastIns | dd bs=64k of=/dev/rmt0
```

To back up a database named `origDSN` to the directory `/users/rob/tmp` and restore it to the database named `restoredDSN`, use:

```
ttBackup -type fileFull -dir /users/rob/tmp -fname restored origDSN
ttRestore -dir /users/rob/tmp -fname restored restoredDSN
```

Notes

When an incremental backup has been enabled, TimesTen creates a backup hold in the transaction log file. This hold can be seen using the `ttLogHolds` built-in procedure. The backup hold is used to determine which log records should be backed up upon subsequent incremental backups. Only changes since the last incremental backup are updated. A side effect to creating the backup hold is that it prevents transaction log files from being purged upon a checkpoint operation until the hold is advanced by performing another incremental backup or removed by disabling incremental backups.

Transactions that committed after the most recent checkpoint are not reflected in the backup.

Up to one checkpoint and one backup may be active at the same time, with these limitations:

- A backup never needs to wait for a checkpoint to complete.
- A backup may need to wait for another backup to complete.
- A checkpoint may need to wait for a backup to complete.

This utility is supported only where the TimesTen Data Manager is installed.

You cannot back up temporary databases.

See also

`ttBulkCp`
`ttMigrate`
`ttRestore`

ttBulkCp

Description

Copies data between TimesTen tables and ASCII files. `ttBulkCp` has two modes:

- In *`copy-in mode`* rows are copied into an existing TimesTen table from one or more ASCII files (or `stdin`).
- In *`copy-out mode`* an entire TimesTen table is copied to a single ASCII output file (or `stdout`).

UNIX

On UNIX, this utility is supported for TimesTen Data Manager DSNs. For Client DSNs, use the utility `ttBulkCpCS`.

This utility only copies out the objects owned by the user executing the utility, and those objects for which the owner has `SELECT` privileges. If the owner executing the utility has the `ADMIN` privilege, `ttBulkCP` copies out all objects.

Required privilege

This utility requires the `INSERT` privilege on the tables it copies information into. It requires the `SELECT` privilege on the tables it copies information from. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttBulkCp {-h | -help | -? | -helpfull}

ttBulkCp {-V | -version}

ttBulkCp -i [-cp numTrans | final] [-d errLevel]
[-e errorFile] [-m maxErrs] [-sc] [-t errLevel]
[-u errLevel] [-v 0|1] [-xp numRows | rollback]
[-Cc | -Cnone] [-tformat timeFormat] [-dateMode dateMode]
[-tsformat timeStampFormat] [-dformat | -D dateFormat]
[-F firstRow] [-L lastRow] [-N ncharEncoding] [-Q 0|1]
[-S errLevel] {-connStr connection_string | DSN}
[owner.]tableName [dataFile ...]

ttBulkCp -o [-sc] [-v 0|1] [-A 0|1] [-Cc | -Cnone]
[-tformat timeFormat] [-tsformat timeStampFormat]
[-dateMode dateMode] [-dformat | -D dateFormat]
[-N ncharEncoding] [-noForceSerializable | -forceSerializable]
[-tsprec precision] [-Q 0|1]
{-connStr connection_string | DSN} [owner.]tblName
[dataFile]
```

Options

`ttBulkCp` has the options:

Option	Description
`-Cnone` `-Cc`	`-Cnone` disables the use of comments in the output file.`-Cc` sets the default comment character to c. If no default comment character is specified, the pound character (#) is used. The `-C` option takes the values: `\t` (tab) or any of the characters:`~ ! @ # % ^ & * ( ) = : ; | < > ? , /` This option overrides the `COMMENTCHAR` file attribute.
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the database to be copied.
`-D | -dformat` *`dateFormat`*	Sets the date format. Legal fixed values are described in "Date, time and timestamp values" on page 3-22. This option overrides the `DFORMAT` file attribute. The default is `ODBC`. See also `-tformat` and `-tsformat`.
`dataFile`	For copy-in mode, specifies the path name(s) of one or more ASCII files containing rows to be inserted into the table. If no files are given, the standard input is used. A single hyphen (-) is understood to mean the standard input.For copy-out mode, specifies the path name of the file into which rows should be copied. If no file is given, the standard output is used. A single hyphen (-) is understood to mean the standard output.
`-dateMode` *`dateMode`*	Specifies whether `ttBulkCp` treats an Oracle DATE type as a simple date (without hour, minute and second fields) or as a timestamp (with hour, minute and second fields). For copy-in mode, the default behavior for input is date. For copy-out mode, the default behavior for output is timestamp. TimesTen truncates the data and issues a warning if you select `-dateMode date` in output mode and one or more date columns have a time component that is not 12:00:00 am. This option overrides the `DATEMODE` file attribute.
`-forceSerializable` `-noForceSerializable`	The `-forceSerializable` option indicates that `ttBulkCp` should use serializable isolation regardless of the DSN or connection string settings. This is the default behavior. `-noForceSerializable` indicates that `ttBulkCp` should honor the isolation level in the DSN or connection string. If you specify the `-noForceSerializable` option and the DSN or connection string indicates a non-serializable isolation mode, a warning is included in the output: `Warning: This output was produced using a non-serializable isolation level. It may therefore not reflect a transaction-consistent state of the table.` For more information on isolation modes, see "Transaction isolation levels" in the *Oracle TimesTen In-Memory Database Operations Guide*.
`-h -help` `-?`	Prints a short usage message and exits.
`-helpfull`	Prints a longer usage message and exits.
`-i`	Selects copy-in mode.

Option	Description
`-m` *`maxErrors`*	Maximum number of errors to report. Default is 10; a few extra related errors may be reported. If 0, the utility only connects, then returns.
`-N` *`ncharEncoding`*	Specifies the input and output character encoding for NCHAR types. Valid values are UTF8, UTF-8 or ASCII.
`-o`	Selects copy-out mode.
`owner`	Specifies the owner of the table to be saved or loaded. If owner is omitted, TimesTen looks for the table under the user's name and then under the user name `SYS`. This parameter is case-insensitive.
`-Q [0 | 1]`	Indicates whether character-string values should be enclosed in double quotes ■ `0` - indicates that strings should not be quoted ■ `1` - (the default) indicates that strings should be quoted.This option overrides the `QUOTES` file attribute.
`-s` *`c`*	Sets the default field-separator character to *`c`*. If no default field-separator is specified, a comma (,) is used. The `-s` option takes the values `\t` (tab) or any of the characters:`~ ! @ # % ^ & * ( ) = : ; | < > ? , /` This option overrides the `FSEP` file attribute.f
`tableName`	Specifies the name of the table to be saved or loaded. This parameter is case-insensitive.
`-tformat` *`timeFormat`*	Sets the time format. Legal values are defined in "Date, time and timestamp values" on page 3-22. The default value is `ODBC`. This option overrides the `TSFORMAT` file attribute. See also `-D | -dformat` and `-tsformat`.
`-tsformat` *`timestampFormat`*	Sets the timestamp format. Legal fixed values are described in "Date, time and timestamp values" on page 3-22. The default value is `DF*TF+FF`, which is the concatenation of the date format, the time format and fractional seconds. This option overrides the `TFORMAT` file attribute. See also `-D | -dformat` and `-tformat`.
`-V | -version`	Prints the release number of `ttBulkCp` and exits.
`-v [0 | 1]`	Sets the verbosity level. `0` - suppresses the summary. `1` - (the default) prints a summary of rows copied upon completion.

The following options can be used in copy-out (`-o`) mode only. You must have SELECT privileges on the specified tables.

Option	Description
`-A [0 | 1]`	Indicates whether `ttBulkCp` should suppress attribute lines in the output file. ■ `0` - (the default) `ttBulkCp` may write attribute lines into the output file ■ `1` - suppresses output of attribute lines.

Option	Description
`-tsprec` *`precision`*	When used with the `-o` option, truncates timestamp values to precision. `ttBulkCp` allows up to 6 digits in the fraction of a second field. Truncation may be necessary when copying timestamps using other RDBMS.

The following options can be used in copy-in (`-i`) mode only. You must have INSERT privileges on the specified tables.

Option	Description
`-cp` *`numTrans`* `-cp final`	Sets the checkpoint policy for the copy in. A value of 0 indicates that `ttBulkCp` should never checkpoint the database, even after the entire copy is complete. A nonzero value indicates that `ttBulkCp` should checkpoint the database after every numTrans transactions, and again after the entire load is complete. A value of `final` indicates that `ttBulkCp` should checkpoint the database only when the entire copy is complete. The default value is 0. Periodic checkpoints can only be enabled if periodic commits are also enabled. See the `-xp` option.

`-d error`
`-d warn`
`-d ignore`
	By default, `ttBulkCp` does not consider rows that are rejected because of constraint violations in a unique column or index to be errors.
	`-d error-` specifies that constraint violations should be considered errors. Duplicate rows are then counted against maxErrs (see `-m`) and placed into the error file (see `-e`).
	`-d warn-` specifies that `ttBulkCp` should copy the offending rows into the error file but should not count them as errors.
	`-d ignore-` (the default) specifies that `ttBulkCp` should silently ignore duplicate rows.
	Regardless of the setting of `-d`, the duplicate rows are not inserted into the table.

`-e` *`errFile`*	Indicates the name of the file where `ttBulkCp` should place information about rows that cannot be copied into the TimesTen table because of errors. These errors include parsing errors, type-conversion errors and constraint violations. The value of *`errFile`* defaults to `stderr`. The format of the error file is the same as the format of the input file (see "datafile format" on page 3-18), so it should be possible to correct the errors in the error file and use the corrected error file as an input file for a subsequent run of `ttBulkCp`.
`-F` *`firstRow`*	Indicates the number of the first row that should be copied. This option can be used (perhaps with `-L`) to copy a subset of rows into the TimesTen table. Rows are numbered starting at 1. If more than one input file is specified, rows are numbered consecutively throughout all the files. The default value is 1.
`-L` *`lastRow`*	Indicates the number of the last row that should be copied. See the description of `-F`. A value of 0 specifies the last row of the last input file. The default value is 0.

Option	Description
`-S` error `-S` warn `-S` ignore	By default, `ttBulkCp` issues an error when it encounters a value that exceeds its maximum scale. This error can be generated for a decimal value whose scale exceeds the maximum scale of its column or for a `TIMESTAMP` value with more than 6 decimal places of fractional seconds (i.e., sub-microsecond granularity). `-S error` - (the default) specifies that `ttBulkCp` should not insert a row containing a value that exceeds its maximum scale into the table and that it should place an error into the error file. `-S warn` - specifies that `ttBulkCp` should right-truncate the value to its maximum scale before inserting the row into the table and that it should place a warning into the error file. `-S ignore` - specifies that `ttBulkCp` should silently right-truncate the value to its maximum scale before inserting the row into the table.

`-t error`
`-t warn`
`-t ignore`
	By default, `ttBulkCp` issues an error when a `CHAR`, `VARCHAR2`, `NCHAR`, `NVARCHAR2`, `BINARY` or `VARBINARY` value is longer than its maximum column width.
	`-t error` - (the default) specifies that rows containing long string or binary attributes should not be inserted into the TimesTen table and that an error should be placed into the error file.
	`-t warn` - specifies that long string or binary attributes should be truncated to the maximum column length before being inserted into the table but that a warning should be placed into the error file.
	`-t ignore` - specifies that long string or binary attributes should be silently truncated to the maximum column length before being inserted into the table.

`-u error`
`-u warn`
`-u ignore`
	By default, `ttBulkCp` issues an error when a real, float or double attribute underflows. Underflow occurs when a floating point number is so small that it is rounded to zero.
	`-u error` - (the default) specifies that rows containing a real, float or double value that underflow should not be inserted into the TimesTen table and that an error should be placed into the error file.
	`-u warn` - specifies that 0.0 should be inserted for real, float or double attributes that underflow, but that a warning should be placed into the error file.
	`-u ignore` - specifies that 0.0 should be silently inserted for real, float or double attributes that underflow.

`-xp` *`numRows`* `-xp rollback`	Sets the transaction policy for the load. A value of 0 indicates that `ttBulkCp` should perform the entire load as a single transaction and should commit that transaction whether the load succeeds or fails. A value of `rollback` indicates that `ttBulkCp` should perform the entire load as a single transaction and should roll that transaction back if the load fails. A nonzero value indicates that `ttBulkCp` should commit after every *`numRows`* successful inserts. The default value is 1000. The `-xp` option can be used with `-cp` option to enable periodic checkpointing of the database.

datafile format

Every line of a `ttBulkCp` input file is one of the following: a blank line, a comment line, an attribute line or a data line.

- Blank lines are lines with no characters at all, including whitespace characters (space and tab). Blank lines are ignored by `ttBulkCp`.
- Comment lines begin with the comment character. The default comment character is #; this default can be overridden with the `-C` command-line option or the `COMMENTCHAR` file attribute (see "Attribute line format" on page 3-19). The comment character must be the first character on the line. Comment lines are ignored by `ttBulkCp`. Comments at the end of data lines are not supported.
- Attribute lines are used for setting file attributes, which control the formatting of the datafile. Attribute lines begin with the ten-character sequence `##ttBulkCp`. The full syntax for attribute lines is described in "Attribute line format" on page 3-19. Attribute lines can appear anywhere in the datafile.
- Data lines contain the rows of the table being copied. Data lines in the datafile and rows of the table correspond one-to-one; that is, each data line completely describes exactly one row. Each data line consists of a list of column values separated by the field separator character. The default field separator is a comma (,). This default can be overridden by the `-s` command-line option or the `FSEP` file attribute. The full syntax for data lines is described in "Data line format" on page 3-20.

Attribute line format

The format of an attribute line is:

```
##ttBulkCp[:attribute=value]...
```

Attribute lines always begin with the ten-character sequence `##ttBulkCp`, even if the comment character is not `#`. This sequence is followed by zero or more file attribute settings, each preceded by a colon.

Attribute settings remain in effect until the end of the input file or until they are changed by another attribute line in the same input file. The values of any file attributes that are omitted in an attribute line are left unchanged.

Command line options take precedence over the values in the file attributes that are supported by `ttBulkCp`. Those file attributes are:

- `VERSION`: Specifies the version of the file format used in the file, expressed as *`major.minor`*. The only supported version is 1.0.
- `DATEMODE`: Specifies whether an Oracle `DATE` type is specified as simple date or as timestamp.
- `FSEP`: Specifies the field separator character used in the file. The field separator can be set to `\t` (tab) or any of the characters: `~ ! @ # $ % ^ & * ( ) = : ; | < > ? , /`
- `QUOTES`: Indicates whether character string values in the file are enclosed in double quotes. The value can be 0, to indicate that strings are not quoted, or 1, to indicate that strings are quoted. This value can be overridden with the `-Q` option.
- `COMMENTCHAR`: Specifies the comment character used in the file. The comment character can be set to \t (tab) or any of the characters: `~ ! @ # $ % ^ & * ( ) = : ; | < > ? , /`

The comment character can also be set to the value `none`, which disables the use of comments in the datafile.

- `DFORMAT`: Sets the date format. Legal values are described in "Date, time and timestamp values" on page 3-22. When a custom format is used, it should be

enclosed in single quotes. This value can be overridden with the `-D/-dformat` command-line option. See also `TFORMAT` and `TSFORMAT`.

- `NCHARENCODING`: Indicates the encoding to be used for the NCHAR and NVARCHAR2 data types. The value may be either ASCII or UTF-8.
- `TFORMAT`: Indicates the time format. Legal values are described in "Date, time and timestamp values" on page 3-22. When a custom format is used, it should be enclosed in single quotes. This value can be overridden with the `-tformat` command-line option. See also `DFORMAT` and `TSFORMAT`.
- `TSFORMAT`: Sets the timestamp format. Legal values are described in "Date, time and timestamp values" on page 3-22. When a custom format is used, it should be enclosed in single quotes. This value can be overridden with the `-tsformat` command-line option. See also `DFORMAT` and `TFORMAT`.

Examples

The following header line sets the field separator character to `$` and disables quoting of character strings:

```
##ttBulkCp:FSEP=$:QUOTES=0
```

The following header line disables comments and sets the date format to the Oracle format:

```
##ttBulkCp:COMMENTCHAR=none:DFORMAT=Oracle
```

The following header line set the date format to a custom format:

```
##ttBulkCp:DFORMAT='Mon DD, YYYY'
```

Data line format

Data lines contain the row data of the table being copied. Each data line corresponds to a row of the table; rows cannot span input-file lines. A data line consists of a list of column values separated by the field separator character. Unnecessary whitespace characters should not be placed either before or after the field separator. The format of each value is determined by its type.

NULL values

`NULL` values can either be expressed as `NULL` (all capitals, no quotes) or as empty fields.

Character and unicode strings

`CHAR`, `VARCHAR2`, `NCHAR`, `NVARCHAR2`: If quoting of character strings is enabled (the default), then strings and characters must be enclosed in double quotes. If quoting of character strings is disabled, then any double-quote characters in the string are considered to be part of the string itself. `ttBulkCp` recognizes the following backslash-escapes inside a character string, regardless of whether quoting of strings is enabled:

- `\"` The double-quote character. If character-string quoting is enabled, then all double quote characters in the string must be escaped with a backslash. If character-string quoting is disabled, then it is permissible, but not necessary, to use the backslash.
- `\t` The tab character.
- `\n` The newline character.

- `\r` The carriage return character.
- `\\` The backslash character.
- `\xyz` (CHAR and VARCHAR2 only) The character whose ASCII value is *xyz*, where *xyz* is a three-character octal number, as in `\033`.
- `\uxyzw` (NCHAR and NVARCHAR2 only) The character whose unicode value is *xyzw*, where *xyzw* is a four-digit hexadecimal number, as in`\ufe4a`. The `\uxyzw` notation is supported in both UTF-8 and ASCII encoding modes.

In addition, any of the `~ ! @ # $ % ^ & * ( ) = : ; | < > ? , /` characters can be escaped with a backslash. Although it is unnecessary to escape these characters in most cases, doing so prevents them from being mistaken for a comment character or a field separator when character-string quoting is disabled.

If character-string quoting is enabled, the empty string (represented as `" "`) is distinct from NULL. If character-string quoting is disabled, then empty strings cannot be represented, as they cannot be distinguished from NULL.

For unicode strings, unicode characters encoded using `UTF-8` multibyte sequences are supported in the `UTF-8` encoding mode only. If these sequences are used with the `ASCII` encoding mode, `ttBulkCp` interprets each byte in the sequence as a separate character.

For fixed-length `CHAR` and `NCHAR` fields, strings that are shorter than the field length are padded with blanks. For `VARCHAR2` and `NVARCHAR2` fields, the string is entered into TimesTen exactly as given in the datafile. Trailing blanks are neither added nor removed.

Binary values

`BINARY`, `VARBINARY`: If quoting of character strings is enabled (the default), binary values are delimited by curly braces (`{...}`). If quoting of character strings is disabled, then curly braces should not be used. Whether character-string quoting is enabled or disabled, binary values may start with an optional `0x` or `0X`.

Each byte of binary data is expressed as two hexadecimal digits. For example, the four-byte binary string:

```
01101000 11001010 01001001 11101111
```

would be expressed as the eight-character hexadecimal string:

```
68CA49EF
```

Digits represented by the letters A through F can either be upper- or lower-case. The hexadecimal string cannot contain white spaces. Because each pair of characters in the hexadecimalstring is converted to a single binary byte, the hexadecimalstring must contain an even number of characters. For fixed-length binary fields, if the given value is shorter than the column length, the value is padded with zeros on the right. For varbinary values, the binary value is inserted into TimesTen exactly as given in the datafile.

If character-string quoting is enabled, a zero-length binary value (represented as { }) is distinct from NULL. If character-string quoting is disabled, then zero-length binary values cannot be represented, as they cannot be distinguished from NULL.

Integer values

TINYINT, SMALLINT, INTEGER, BIGINT: Integer values consist of an optional sign followed by one or more digits. Integer values may not use E-notation. Examples:

```
-14 98765 +186
```

Floating-point values

REAL, FLOAT, DOUBLE: Floating-point values can be expressed with or without decimal points and may use E-notation. Examples:

```
3.1415
-0.00004
1.1e-3
5e3
.56
-682
-.62E-4
170.
```

Fixed-point values

DECIMAL, NUMERIC: Decimal values can be expressed with or without decimal points. Decimal values may not use E-notation. Examples:

```
5
-19.5
-11
000
-.1234
45.
-57.0
0.8888
```

Inf, -Inf and NaN values

Inf, -Inf and Nan values: Infinity and Not a Number values can be represented as strings to represent the corresponding constant value (all are case insensitive):

String	Value
NAN	NaN
[+]INF	Inf
-INF	-Inf

TimesTen outputs the values as: NAN, INF and -Inf.

Date, time and timestamp values

Formats for date, time and timestamp values can be specified either by selecting a fixed datetime format or by defining a custom datetime format. The custom datetime formats are defined using format specifiers similar to those used by the TO_DATE and TO_CHAR SQL functions, as described in the following table.

In many cases, it is not necessary to define the timestamp format, even when a custom date or time format is used, because the default TimesTen format (DF*TF+FF) is defined in terms of the date and time formats. Therefore, setting the date format sets not only the format for date values, but also for the date portion of timestamp values. Similarly, setting the timestamp format affects both time values and the time portion of the timestamp values.

Specifier	Descriptions and restrictions
Q	Quarter. Cannot be used in copy-in mode.
YYYY	Year (four digits).

Specifier	Descriptions and restrictions
Y,YYY	Year (with comma as shown)
YYY	Year (last three digits). Cannot be used in copy-in mode.
Y	Year (last digit). Cannot be used in copy-in mode.
MONTH	Month (full name, blank-padded to 9 characters, case-insensitive).
MON	Month (three character prefix, case-insensitive).
MM	Month (01 through 12)
DD	Day of the month (01 through 31)
HH24	Hour (00 through 23)
HH12	Hour (01 through 12).Must be used with AM/PM for copy-in mode.
HH	Hour (01 through 12).Must be used with AM/PM for copy-in mode.
MI	Minute (00 through 59)
SS	Second (00 through 59)
FF	Fractional seconds.Six digits, unless overridden with the `-tsprec` option.
FF*n*	Fractional seconds (number of digits specified by *n*).
+FF	In copy-in mode, matches, optional decimal point plus one or more fractional seconds. In copy-out mode, same as `.FF`
+FF*n*	In copy-in mode, same as `+FF`. In copy-out mode, same as `.FF`*n*
AM PM	Meridian indicator without dots. In copy-in mode, this must be used with HH or HH12, but not HH24.
A.M. P.M.	Meridian indicator with dots. In copy-in mode, this must be used with HH or HH12, but not HH24.
DF	Current date format (can only be used in timestamp format).
TF	Current time format (can only be used in timestamp format).
- / ; :	Punctuation that are matched in copy-in mode or output in copy-out mode.
"text"	Text that is matched in input mode or output in copy-out mode.
*	Matches 0 or more whitespace characters (space or tab) in copy-in mode or outputs 1 space in copy-out mode.

Fixed, date, time and timestamp formats

For date values, the fixed formats are

Format	Description
ODBC	YYYY-MM-DD Example: `1997-01-03` (default value)
Oracle	DD-Mon-YYYY Example: `03-Jan-1997`
SYBASE1	MM/DD/YYYY Example: `01/03/1997`

Format	Description
SYBASE2	DD-MM-YYYY Example: `03-01-1997`
SYBASE3	Mon*DD*YYYY Example: `Jan 03 1997`

For time values, the only fixed format is ODBC:

Format	Description
ODBC	HH24:MI:SS Example: `07:47:23`

For timestamp values, the fixed formats are:

Format	Description
ODBC	YYYY-MM-DD*HH24:MI:SS+FF Example: `1997-01-03 07:47:23`
Oracle	DD-Mon-YYYY*HH24:MI:SS+FF Example: 03-Jan-1997 `07:47:23`
SYBASE1	MM/DD/YYYY*HH24:MI:SS+FF Example: 01/03/1997 `07:47:23`
SYBASE2	DD-MM-YYYY*HH24:MI:SS+FF Example: 03-01-1997 `07:47:23`
SYBASE3	Mon*DD*YYYY*HH24:MI:SS+FF Example: Jan 03 1997`07:47:23`

The default timestamp value is: `'DF*TF+FF'`

Examples

The following input file is for a table with five columns: two char columns, a double column, an integer column and a varbinary column. In the "Mountain View" line, the last three columns have NULL values.

```
##ttBulkCp
# This is a comment.
###### So is this.
# The following line is a blank line.

"New York","New York",-345.09,12,{12EF87A4E5}
"Milan","Italy",0,0,{0x458F}
"Paris","France",1.4E12,NULL,{F009}
"Tokyo","Japan",-4.5E-18,26,{0x00}
"Mountain View","California",,,
```

Here is an equivalent input file in which quotes are disabled, the comment character is '$' and the field separator is '|':

```
##ttBulkCp:QUOTES=0:COMMENTCHAR=$:FSEP=|
$ This is a comment.
```

```
$$$$$$ So is this.
$ The following line is a blank line.

New York|New York|-345.09|12|12EF87A4E5
Milan|Italy|0|0|0x458F
Paris|France|1.4E12|NULL|F009
Tokyo|Japan|-4.5E-18|26|0x00
Mountain View|California|||
```

The following command dumps the contents of table `mytbl` from database `mystore` into a file called `mytbl.dump`.

```
ttBulkCp -o DSN=mystore mytbl mytbl.dump
```

The following command loads the rows listed in file `mytbl.dump` into a table called `mytbl` on database `mystore`, placing any error messages into the file `mytbl.err`.

```
ttBulkCp -i -e mytbl.err DSN=mystore mytbl mytbl.dump
```

The above command terminates after the first error occurs. To force the copy to continue until the end of the input file (or a fatal error), use `-m 0`, as in:

```
ttBulkCp -i -e mytbl.err -m 0 DSN=mystore mytbl mytbl.dump
```

To ignore errors caused by constraint violations, use `-d ignore`:

```
ttBulkCp -i -e mytbl.err -d ignore DSN=mystore mytbl mytbl.dump
```

Notes

ttBulkCp explicitly sets the `Overwrite` database attribute to 0, to prevent accidental destruction of a database. For more information, see "Overwrite" on page 1-39.

Real, float or double values may be rounded to zero when the floating point number is small.

When specifying date, time and timestamp formats, incomplete or redundant formats are not allowed in input mode. Specifiers that reference fields that are not present in the data type (for example a minute specifier in a date format) return errors in copy-out mode. In copy-in mode, the values of those specifiers are ignored.

The following caveats apply when disabling quoted strings in the `ttBulkCp` datafile:

- Empty strings and zero-length binary values cannot be expressed, as they cannot be distinguished from `NULL`.
- If the field separator character appears inside a character string, it must be escaped with a backslash or else it is treated as an actual field separator.
- If a data line begins with a character string and that string begins with the comment character, that character must be escaped with a backslash or else the line is treated as a comment. Setting the comment character to `none` can prevent this, as long as there are no actual comments in the file.

For UTF-8, NCHAR are converted to UTF-8 encoding and then output. UTF-8 input is converted to NCHAR.

For ASCII, those NCHAR values that correspond to ASCII characters are output as ASCII. For those NCHAR values outside of the ASCII range, the escaped Unicode format is used.

This utility is for use specifically with TimesTen tables. It is not supported with passthrough to Oracle.

WINDOWS

On Windows, this utility is supported for all TimesTen Data Manager and Client DSNs.

See also

ttBackup
ttMigrate
ttRestore

ttCapture

Description

Captures information about the state of TimesTen at the time the command is used. This information may be useful in diagnosing problems. Sometimes TimesTen Customer Support must make repeated incremental requests for information to diagnose a customer's problem in the field.

The information captured by this utility may be requested by TimesTen Customer Support and may be sent with your support email.

The utility does not interpret errors. It only collects information about the state of things and sends output to the `ttcapture.`*`date.number`*`.log` file in the directory from which you invoke the `ttCapture` utility. This utility collects general information that is usually relevant to support cases.

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

Required privilege

This utility requires the instance administrator privilege.

If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttCapture {-h | -help | -?}
ttCapture {-V | -version}
ttCapture [-noinstinfo] [-nosysinfo] [-stdout | -dest dir] [-logdir dir]
 [dspath |DSN]
```

Options

`ttCapture` has the options:

Option	Description
`-dest` *`dir`*	Writes the output file to the designated directory.
`DSN`	Specifies an ODBC data source name of the database to be checked.
`dspath`	The fully qualified name of the database to be evaluated. This is not the DSN associated with the connection but the fully qualified database path name associated with the database as specified in the `DataStore=` parameter of the database's ODBC definition. For example, for a database consisting of files `/home/payroll/1997.ds0`, `/home/payroll/1997.ds1`, and several transaction log files `/home/payroll/1997.log`*`n`*, *`dspath`* is `/home/payroll/1997`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

Option	Description
`-logdir` *`dir`*	Specifies the location of the log directory. Must be used with the `-dsname` option. If not specified, the log directory may not be available.
`-noinstinfo`	Do not capture any installation information.
`-nosysinfo`	Do not capture any system information.
`-stdout`	On UNIX systems, writes all output to `stdout`, instead of writing the output to a file.
`-V | -version`	Prints the release number of `ttCapture` and exits.

Examples

To capture data on the `test_db` database and write the database checkpoint files to the directory `D:\my_data\recover\test_db`, use:

```
ttCapture -dest "D:\my_data\recover\test_db" test_db
```

Note

This utility is supported only where the TimesTen Data Manager is installed.

ttCheck

Description

Performs internal consistency checking within a TimesTen database. You can specify a specific structure to be checked and a desired level of checking.

Required privilege

This utility requires the ADMIN privilege.

If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttCheck {-h | -help | -?}
ttCheck {-V | -version}
ttCheck [ [-blkDir] [-compHeap] [-header] [-heap] [-indexHeap] [-log]
[-permBlkDir] [-permHeap] [-tempBlkDir] [-tmpHeap]
[-tables tblName [...]] [-users userName [...]]
[-level levelNum] ] [...]
[-m maxErrors] [-f outFile] [-v verbosity]
{DSN | [-connstr] connection_string | dspath}
```

Options

`ttCheck` has the options:

Option	Description
`-blkDir`	Checks all the block directories.
`-compHeap`	Checks the compilation heap structure.
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the database to be checked.
`dspath`	The fully qualified name of the database to be checked. This is not the DSN associated with the connection. It is the fully qualified database path name associated with the database as specified in the `DataStore=` parameter of the database's ODBC definition. For example, for a database consisting of files `/home/payroll/1997.ds0`, `/home/payroll/1997.ds1`, and several transaction log files `/home/payroll/1997.log`n, *`dspath`* is `/home/payroll/1997`.
`-f` *`outFile`*	Specifies the output file name; defaults to `stdout`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-header`	Checks the content of the database header.
`-heap`	Checks all heap structures.
`-indexHeap`	Checks the index heap structure.

Option	Description
`-level` *`levelNum`*	Can be used to indicate the level of checking for header, block directory, heap and table. Different structures can be checked using different levels in a same command. A level specification is applied to all structures specified to its left in the command string that do not have a level specification. A level specification is applied to all structures if no structure is specified in the command string. 1-Checks sanity bytes and simple fields. For example, counts enums for validity in all high-level structures. 2-Does all checks in level 1, plus checks the validity of structures, referenced by fields in other structures. 3-Does all checks in level 2, plus checks each table row for column values. For example, checks valid VARCHAR2 and FLOAT sizes. 4-(the default) Does all checks in level 3, plus checks index/table mapping for each row and each index.
`-log`	Checks the log buffer.
`-m` *`maxErrors`*	Maximum number of errors to report. Default is 10; a few extra related errors may be reported. If 0, the utility only connects, then returns.
`-permBlkDir`	Checks the permanent partition block directory.
`-permHeap`	Checks the permanent heap structure.
`-tables` *`tblName`* `[...]`	Checks table(s) specified by *tblName*.
`-tempBlkDir`	Checks the temporary partition block directory.
`-tmpHeap`	Checks the temporary heap structure.
`-users` *`userName`* `[...]`	Checks tables belonging to the user(s) specified by *userName*.
`-V | -version`	Prints the release number of `ttCheck` and exits.
`-v` *`verbosity`*	0-no output (program's exit status indicates if an error was found). 1-(the default) enable error output only. 2-error output and a progress report.

Examples

To perform a check of all structures in the `test_db` database, use:

```
ttCheck test_db
```

To perform a sanity check of all structures in the `test_db` database, use:

```
ttCheck -level 1 test_db
```

To perform a check of all tables in the test_db database, use:

```
ttCheck -tables test_db
```

To check the physical structures and row contents of all tables in the `test_db` database, use:

```
ttCheck -tables -level 3 test_db
```

To perform a sanity check of all heap structures, row contents and indexes of all tables in the `test_db` database, use:

```
ttCheck -heap -level 1 -tables -level 4 test_db
```

To check the physical structures and row contents of tables `tab1` and `tab2` in the `test_db` database, use:

```
ttCheck -tables tab1 tab2 -level 3 test_db
```

Notes

While primarily intended for use by TimesTen customer support to diagnose problems with internal data structures of a TimesTen database, the information returned by `ttCheck` may be useful to system administrators and developers.

The `ttCheck` utility should be run when there are no active transactions on the system. If run on a shared database and other transactions are active, `ttCheck` may return errors when the database is in fact intact.

The `ttCheck` utility checks views in the same manner as other tables in a database. The utility cannot verify that the contents of a view matches view query's result.

If no structures are specified, `ttCheck` checks all structures. No errors are returned if a specified table's name or user is not found.

This utility may take some time to run. Verbosity level 2 allows you to print a progress report.

This utility is supported only where the TimesTen Data Manager is installed.

ttCWAdmin

Description

Manages TimesTen active standby pairs that take advantage of the high availability framework of Oracle Clusterware. This utility starts administrative processes, generates scripts and performs other functions to administer active standby pairs and the corresponding Clusterware resources.

For more information about using Oracle Clusterware to manage TimesTen active standby pairs, see *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*.

These commands are available only with advanced high availability:

- `ttCWAdmin -addMasterHosts`
- `ttCWAdmin -addSubscriberHosts`
- `ttCWAdmin -createVIPs`
- `ttCWAdmin -delMasterHosts`
- `ttCWAdmin -delSubscriberHosts`
- `ttCWAdmin -ocrConfig`
- `ttCWAdmin -dropVIPs`

These commands fail with basic high availability:

Required privilege

On Windows 2008, a user with Administrators privileges can execute all commands as Administrator. On other supported Windows platforms, any user that has Administrator privileges can execute all commands in this utility.

On UNIX, the `root` user can execute all commands in this utility. These commands must be executed by the `root` user:

- `ttCWAdmin -addMasterHosts`
- `ttCWAdmin -addSubscriberHosts`
- `ttCWAdmin -createVIPs`
- `ttCWAdmin -delMasterHosts`
- `ttCWAdmin -delSubscriberHosts`
- `ttCWAdmin -ocrConfig`
- `ttCWAdmin -dropVIPs`

The admin user can execute all other commands in this utility.

If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttCWAdmin {-h | -help | -?}

ttCWAdmin {-V | -version}
```

```
ttCWAdmin -init [-hosts "host_name1, host_name2[, ...]"]

ttCWAdmin {-createVIPs | -dropVIPs | -create | -drop | -restore | -start |
           -stop | -status} [-ttclusterini path] -dsn DSN

ttCWAdmin -switch -dsn DSN

ttCWAdmin -relocate -dsn DSN

ttCWAdmin -ocrConfig

ttCWAdmin -addMasterHosts [-hosts "host_name1, host_name2[, ...]"] -dsn DSN

ttCWAdmin -delMasterHosts [-hosts "host_name1, host_name2[, ...]"] -dsn DSN

ttCWAdmin -addSubscriberHosts [-hosts "host_name1, host_name2[, ...]"] -dsn DSN

ttCWAdmin -delSubscriberHosts [-hosts "host_name1, host_name2[, ...]"] -dsn DSN

ttCWAdmin -shutdown [-hosts "host_name1, host_name2[, ...]"]
```

Options

`ttCWAdmin` has these general options:

Option	Description
`-addMasterHosts`	Adds spare hosts to the poll of master hosts dynamically, when high availability is employed. On the command line, separate multiple host names by commas. On UNIX systems, only the `root` user can execute this command.
`-addSubscriberHosts`	Adds spare hosts to the poll of subscriber hosts dynamically, when high availability is employed. On the command line, separate multiple host names by commas. On UNIX systems, only the `root` user can execute this command.
`-create`	Creates the active standby pair replication scheme for the specified DSN and creates the associated action scripts. This command prompts for the user ID and password for an internal user with the ADMIN privilege if it does not find this information in the `sys.odbc.ini` file. This command also prompts for a passphrase for the encryption of stored authentication information. This internal user is used to create the active standby pair. If CacheConnect is enabled, the command also prompts for the user password for the Oracle database. The Oracle password is used to set the autorefresh states for cache groups.
`-createVIPs`	Creates virtual IP addresses for the active standby pair.
`-delMasterHosts`	Deletes spare hosts to the poll of master hosts dynamically, when high availability is employed. On the command line, separate multiple host names by commas. The command fails if the indicated hosts are not spare hosts. On UNIX systems, only the `root` user can execute this command.

Option	Description
`-delSubscriberHosts`	Deletes spare hosts to the poll of subscriber hosts dynamically, when high availability is employed. On the command line, separate multiple host names by commas. The command fails if the indicated hosts are not spare hosts. On UNIX systems, only the `root` user can execute this command.
`-drop`	Drops the active standby pair replication scheme and deletes its action scripts.
`-dropVIPs`	Drops the virtual IP addresses for the active standby pair.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-init`	Starts the TimesTen cluster agent.
`-ocrConfig`	(UNIX only.) TimesTen cluster information is stored in the Oracle Cluster Registry (OCR). This option registers the admin user in the OCR. Login as the root user and run this utility from any host in the system, before creating any clusters.
`-relocate`	Relocates the database from the local host to the next available spare host specified in the `MasterHosts` attribute in the configuration file. If no spare host is available, an error is returned. If the database on the local host is active, roles are first reversed so that the remote standby store of the same cluster becomes active. The newly migrated database on the spare host always comes up as the standby database. This is useful to forcefully relocate a database if you must take the host offline, when high availability is employed. This command fails when basic HA is deployed for the same cluster.
`-restore`	Restores the active master database from the backup specified by RepBackupDir. Do not use this command when AutoRecover is enabled.
`-shutdown`	Stops the TimesTen cluster agent.
`-start`	Starts the active standby pair.
`-status`	Obtains the status of resources in the cluster.
`-stop`	Stops the active standby pair.
`-switch`	Reverses the role of an active standby pair in a cluster. The standby database becomes the new active, while the existing active database becomes the standby database.
`-dsn` *`DSN`*	Specifies the DSN for the active standby pair.
`-hosts "`*`host_name1`*`,` *`host_name2`*`[, ...]"`	Specifies the hosts on which to start or shut down the TimesTen cluster agent. If this option is not specified, the TimesTen cluster agent is started or stopped on all hosts.
`-ttclusterini` *`path`*	Specifies the full path name of the `cluster.oracle.ini` file. The default location is in the daemon home directory. The default location is recommended.
`-V | -version`	Prints the release number of `ttCWAdmin` and exits.

Examples

To create and start an active standby pair managed by Oracle Clusterware, using the `clusterDSN` DSN, enter:

```
ttCWAdmin -create -dsn clusterDSN
ttCWAdmin -start -dsn clusterDSN
```

To stop and drop an active standby pair managed by Oracle Clusterware, using the `clusterDSN` DSN, enter:

```
ttCWAdmin -stop -dsn clusterDSN
ttCWAdmin -drop -dsn clusterDSN
```

ttDaemonAdmin

Description

Starts and stops the TimesTen main daemon and Server.

Required privilege

This utility requires the instance administrator privilege.

Syntax

```
ttDaemonAdmin {-h | -help | -?}
ttDaemonAdmin {-V | -version}
ttDaemonAdmin [-force] {-start | -stop | -restart}
ttDaemonAdmin -startserver
ttDaemonAdmin [-force] -stopserver
```

Options

`ttDaemonAdmin` has the options:

Option	Description

`-h`
`-help`
-?
	Prints a usage message and exits.

`-force`	Starts or stops the TimesTen main daemon, even when warnings are returned or with `-stopserver` immediately stops the server processes.
`-restart`	Restarts the TimesTen main daemon.
`-start`	Starts the TimesTen main daemon.
`-startserver`	Starts the TimesTen Server daemon.
`-stop`	Stops the TimesTen main daemon.
`-stopserver`	Stops the TimesTen Server daemon. Without the `-force` option, client/server connections to TimesTen databases are gracefully disconnected after completing any request they may be processing, and then the server exits. With the `-force` option, client/server connections to TimesTen databases are forcefully and immediately terminated, and then the server exits.
`-V | -version`	Prints the release number of `ttDaemonAdmin` and exits.

Notes

Changes to the TimesTen Server options are temporary. To permanently set or disable the TimesTen Server options, you must change the options in the `ttendaemon.options` file.

The `-force` option should be used with caution, as it may leave databases in a state where you must perform recovery procedures.

When you use this utility on Windows Vista, you must be running with Windows Administrative privileges.

When you stop the daemon (`ttDaemonAdmin -stop`), first stop all application connections to the database. This decreases startup time when the daemon is restarted.

If the CRS agent is running, you must stop it on the local host before stopping the TimesTen main daemon (`ttDaemonAdmin -stop`). If you do not stop the CRS agent, the main daemon stops temporarily with this command, but then restarts. To stop the CRS agent, use:

```
ttcwadmin -shutdown -hosts localhost
```

When you use this utility to restart the server, the TimesTen daemon reads the `ttendaemon.options` files to see if it has been changed since it was last read. If the file has been changed, TimesTen checks for the values of the options:

```
-server -serverShmIpc -serverShmSize -noserverlog
```

See also

For a description of all daemon options and instructions for changing the `ttendaemon.options` file, see "Managing TimesTen daemon options" in *Oracle TimesTen In-Memory Database Operations Guide*.

ttDaemonLog

Description

TimesTen uses a TimesTen daemon (referred to as the TimesTen Data Manager Service on Windows) and other background processes, known as subdaemons and agents, to manage access to the either the "user" or "error" log.

By default, TimesTen messages are stored in:

- A user error log that contains information you may need to see. Generally, these messages contain information on actions you may need to take.
- A support log containing everything in the user error log plus information of use by TimesTen Customer Support.

The `ttDaemonLog` utility allows you to control the type of events that are written to and fetched from the TimesTen user and error logs.

There are two versions of the `ttDaemonLog` utility:

- `ttDaemonLog` for Windows
- `ttDaemonLog` for UNIX

Required privilege

This utility requires the instance administrator privilege.

ttDaemonLog for Windows

On Windows, TimesTen user and error log messages are written to the Windows Application Event Log. The `ttDaemonLog` utility controls which events are written to and fetched from the log and displayed to `stdout`.

Syntax

```
ttDaemonLog {-h | -help | -?}
ttDaemonLog {-V | -version}
ttdaemonlog [-show type] [-b | -r | -s] [-f] [-maxlines]
[-loglevel level [DSN | [-connstr] connStr]]
[-[no]logcomponent component [DSN | [-connstr] connStr]]
[-logreset] [-msg messagestring] [-setquiet | -setverbose]
[-n computer]
```

Options

`ttDaemonLog` has the options:

Option	Description
`-b`	Prints all TimesTen-generated syslog entries.
`-f`	When the end of the log is reached, `ttDaemonLog` does not terminate but continues to execute, periodically polling the event log to retrieve and display additional TimesTen log records. This is useful, for example, for generating a display of log data that is updated in real time.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-maxlines`	Maximum number of lines at end of log to display. Defaults to 40 lines if `-f` is specified. If 0, no limit is set
`-logcomponent` *`component`* `-nologcomponent`	Specifies that the given component should be logged. If a DSN or connection string is specified as the component, logging applies only to the specified database. If `-nologcomponent` is specified, no logging is done for the indicated component. Legal values for component are: ■ ALL ■ DAEMON ■ DAEMONDBG ■ REPLICATION ■ ORACLECONNECT
`-loglevel` *`level`*	Specifies that messages of level greater than or equal to the specified level should be logged. If a DSN or connection string is specified as the component, the option applies only to the specified database.
`-logreset`	Resets event logging parameters.
`-msg` *`messagestring`*	Insert string into the TimesTen user log.
`-n` *`computer`*	Displays the log from a different computer. Specify the Universal Naming Convention (UNC) name of the target computer.

Option	Description
`-r`	Print only the TimesTen Replication Agent log. (Same as `-show replication`.)
`-s`	Print only the TimesTen Server log. (Same as `-show server`.)
`-setverbose` `-setquiet`	Enable (`-setverbose`) or disable (`-setquiet`) TimesTen verbose logging.
`-show` *`type`*	Limits output to the given type. Types are: :**all** - Shows all. (default) **replication** - Shows only log entries for replication agents. (Same as `-r` option.) **ora** - Shows only log entries for cache agents **server** - Shows only log entries for TimesTen Server. (Same as `-s` option.)
`-V | -version`	Prints the release number of `ttDaemonLog` and exits.

Examples

By default, the `ttDaemonLog` utility logs messages and errors from all the TimesTen components. You can narrow the scope of what is written to the log by setting the `-nologcomponent` option. The `-nologcomponent` option can be applied to selected databases or all databases.

For example, to prevent messages and errors related to replication for all databases from being written to the log, enter:

```
ttDaemonLog -nologcomponent replication
```

To prevent messages and errors related to replication for the masterdsn database from being written to the log, enter:

```
ttDaemonLog -nologcomponent replication masterdsn
```

If, you want to prevent both replication and IMDB Cache errors and messages from being written, enter:

```
ttDaemonLog -nologcomponent replication
ttDaemonLog -nologcomponent oracleconnect
```

If, after setting a `-nologcomponent` option, you want to re-enable writing errors for a component, you can use the `-logcomponent` option. For example, if after preventing both replication and IMDB Cache errors from being logged, as shown in the example above, you want to re-enable logging of replication errors, enter:

```
ttDaemonLog -logcomponent replication
```

To re-enable logging for all TimesTen components, you can use the `-logreset` option:

```
ttDaemonLog -logreset
```

To display all the output from the TimesTen daemon and server on your local computer, use:

```
ttDaemonLog
```

To display the log output from the host computer named, backup1, use:

```
ttDaemonLog -n backup1
```

To write the log output to the file `C:\TimesTen\logout\log1`, use:

```
ttDaemonLog -file C:\TimesTen\logout\log1
```

The TimesTen Server generates a message each time an application connects to or disconnects from a client DSN if these messages were specified to be generated during installation. To display just the server log messages, use:

```
ttDaemonLog -show server
```

To display just the replication agent messages, use:

```
ttDaemonLog -show replication
```

To display just the cache agent messages, use:

```
ttDaemonLog -show ora
```

To display all messages from the TimesTen processes, use:

```
ttDaemonLog -show all
```

To restore logging to its default "verbose" level, use the `-setverbose` option:

```
ttDaemonLog -setverbose
```

Notes

While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

To permanently set or disable verbose logging, you must change the options in the `ttendaemon.options` file. See "Modifying informational messages" in the *Oracle TimesTen In-Memory Database Operations Guide*.

ttDaemonLog for UNIX

Description

On UNIX, `ttDaemonLog` fetches all TimesTen events from the file generated by `syslogd(1)`. It displays all events to `stdout`.

The TimesTen daemon (`timestend`) records its event log via `syslog(2)`. The eventual disposition of the log information depends on the configuration of your `/etc/syslog.conf` file, which you can customize to log or ignore messages selectively. Messages can be logged into various files depending on the configuration of the file. These files can grow to be quite large. You should prune them periodically to conserve disk space.

Syntax

```
ttDaemonLog {-h | -help | -?}
ttDaemonLog {-V | -version}
ttDaemonLog [-show type] [-b | -r | -s] [-f] [-integer] [-file filename]
 [-facility name] [-loglevel level [DSN | [-connstr] connStr]] [-[no]logcomponent
 component [DSN | [-connstr] connStr]] [-logreset] [-msg string] [-setquiet |
 -setverbose]
```

Options

`ttDaemonLog` has the options:

Option	Description
`-b`	Prints all TimesTen generated syslog entries.
`-f`	When the end of the log is reached, `ttDaemonLog` does not terminate but continues to execute, periodically polling the event log to retrieve and display additional TimesTen log records. This is useful, for example, for generating a display of log data that is updated in real time.
`-facility` *`name`*	Specifies the syslog facility name being used.
`-file` *`filename`*	Specifies the file into which TimesTen is logging messages. If not specified, examine the system's `syslog` configuration to determine where TimesTen messages are being logged.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-integer`	Maximum number of lines at end of log to display. Defaults to 40 lines if `-f` is specified. If 0, no limit is set.

Option	Description
`-logcomponent` *component*	Specifies that the given component should be logged, along with any other components that are already being logged.
`-nologcomponent`	If a DSN or connection string is specified as the component, logging applies only to the specified database. If `-nologcomponent` is specified, no logging is done for the indicated component.Legal values for component are: ■ ALL ■ DAEMON ■ DAEMONDBG ■ REPLICATION ■ ORACLECONNECT
`-loglevel` *level*	Specifies that messages of level greater than or equal to the specified level should be logged. If a DSN or connection string is specified as the component, the option applies only to the specified database.
`-logreset`	Resets event logging parameters.
`-msg` *string*	Insert string into the TimesTen user log.
`-r`	Print only the TimesTen Replication Agent log. (Same as `-show replication`.)
`-s`	Prints the TimesTen Server log. (Same as `-show server`.)
`-setverbose` `-setquiet`	Enable (`-setverbose`) or disable (`-setquiet`) TimesTen verbose logging.
`-show` *type*	Limits output to the given type. Types are: :**all** - Shows all. (default) **replication** - Shows only log entries for replication agents. (Same as `-r` option.) **ora** - Shows only log entries for cache agents **server** - Shows only log entries for TimesTen Server. (Same as `-s` option.)
`-V` | `-version`	Prints the release number of `ttDaemonLog` and exits.

Examples

Except for the example with the `-n` option, all the examples shown under "ttDaemonLog for Windows" on page 3-39 also apply to the UNIX version of `ttDaemonLog`. The following examples, show the use of some of the UNIX-specific options.

To write the log output to the file `/var/adm/syslog/syslog.log`, use:

```
ttDaemonLog -file /var/adm/syslog/syslog.log
```

To direct logging to the `local7` facility, use.

```
ttDaemonLog -facility local7
```

Notes

While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

To permanently set or disable verbose logging, you must change the options in the `ttendaemon.options` file. For information on this file and on configuring `sys.log`, see "Modifying informational messages" in the *Oracle TimesTen In-Memory Database Operations Guide*.

ttDestroy

Description

Destroys a database including all checkpoint files, transaction logs and daemon catalog entries (though not the DSNs).

Required privilege

This utility requires the instance administrator privilege.

Syntax

```
ttDestroy {-h | -help | -?}
ttDestroy {-V | -version}
ttDestroy [[-wait] [-timeout secs]] [-force] {-connStr connection_string |
 DSN | dspath}
```

Options

`ttDestroy` has the options:

Option	Description
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the database to be destroyed.
`dspath`	The fully qualified name of the database to be destroyed. This is not the DSN associated with the connection but the fully qualified database path name associated with the database as specified in the `DataStore=` parameter of the database's ODBC definition. For example, for a database consisting of files `/home/payroll/1997.ds0`, `/home/payroll/1997.ds1`, and several transaction log files `/home/payroll/1997.logn`, *`dspath`* is `/home/payroll/1997`.

`-h`
`-help`
-?
	Prints a usage message and exits.

`-force`	Destroy even if files are from an incompatible version or a different instance of TimesTen.
`-timeout` *`seconds`*	Indicates the time in seconds that `ttDestroy` should wait. If no timeout value is supplied, TimesTen waits five seconds before retrying the destroy operation.
`-V` | `-version`	Prints the release number of `ttDestroy` and exits.
`-wait`	Causes `ttDestroy` to continually retry the destroy operation until it is successful, in those situations where the destroy fails due to some temporary condition, such as when the database is in use.

Example

```
ttDestroy /users/pat/TimesTen/Daily/F112697
```

Notes

Using `ttDestroy` is the only way to delete a database completely and safely. Do not remove database checkpoint or transaction log files manually.

This utility is supported only where the TimesTen Data Manager is installed.

In the case that the database to be destroyed is part of a cache grid, `ttDestroy` performs a detaches the database from the grid.

`ttDestroy` does not perform cleanup of Oracle objects from AUTOREFRESH or AWT cache groups. If there are AUTOREFRESH or AWT cache groups in the database, use the `ttCacheSqlGet` built-in procedure, specifying `NULL` for the cache group name, to generate Oracle SQL to perform cleanup after the database has been destroyed.

ttIsql

Description

You can execute SQL statements and call TimesTen built-in procedures from `ttIsql`. You can execute SQL interactively from the command line. For a detailed description on running SQL from `ttIsql`, use the `-helpfull` option. In addition, you can call a TimesTen built-in procedure with `call` *<procedure-name>*.

On UNIX, this utility is supported for TimesTen Data Manager DSNs. Use `ttIsqlCS` for client/server DSNs.

Required privilege

This utility requires no privileges.

Syntax

```
ttIsql {-h | -help | -? | -helpcmds | - helpfull}
ttIsql {-V | -version}
ttIsql [-f inputFile] [-v verbosity] [-e commands | sql_statement]
[-interactive] [-N ncharEncoding] [-wait] {-connStr connection_string | DSN}
```

Options

`ttIsql` has the options:

Option	Description
`-connStr` *connection_string*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
DSN	Specifies an ODBC data source name of the database to be connected.
`-e` *commands*	Specifies a semi-colon separated list of `ttIsql` commands to execute on start up.
`-f` *filename*	Read SQL commands from *filename*.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-helpcmds`	Prints a short list of the interactive commands.
`-helpfull`	Prints a full description of the interactive commands.
`-interactive`	Forces interactive mode. This is useful when running from an emacs comint buffer.
`-N` *ncharEncoding*	Specifies the character encoding method for `NCHAR` output. Valid values are `LOCALE` or `ASCII`. `LOCALE` (the default) sets the output format to the locale-based setting. If no value is specified, TimesTen uses the system's native language characters.
`-V | -version`	Prints the release number of `ttIsql` and exits.

Option	Description
`-v verbosity`	Specifies the verbosity level. One of: 0 - Shows error information only. If all commands succeed, there is no output. 1 - The basic output generated by commands is displayed. 2 - (the default) Same as level 1, plus it shows more detailed results of commands.At this level simplified SQL error and information messages are displayed. In addition, `ttIsql` commands that are read from an external file are echoed to the display. 3 - Same as level 2, with more detailed error and information messages. 4 - Same as level 3, plus complete error and information messages are displayed. Also displayed are messages about prepared commands, "success" messages for each command that succeeded and content of XLA records.
`-wait`	Waits until successful connect.

Commands

Also see the list of `ttIsql` "Set/show attributes" on page 3-57.

Boolean commands can accept the values "ON" and "OFF" in place of "1" and "0".

`ttIsql` has the commands:

Command	Description
`accept variable[NUM[BER]| CHAR |BINARY_FLOAT | BINARY_DOUBLE] [DEF[AULT]default_value] [PROMPT prompt_text | NOPR[OMPT]] [HIDE]`	Gets input from a user and `DEFINES` the variable. If a type is specified then it will be validated for that type. The default (enclosed in quotes) will be assigned if the user just presses enter. The prompt will be displayed before waiting for input (or can be suppressed). The `HIDE` option stops the terminal from displaying the entered text (for passwords). The prompt is displayed before waiting for input, if specified without the `HIDE` option. The `HIDE` option stops the terminal from displaying the entered text.
`allfunctions [[owner_name_pattern.] table_name_pattern]`	Lists, in a single column, the names of all the PL/SQL functions that match the given pattern selected from `SYS.ALL_OBJECTS`. When a pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists PL/SQL functions matching the pattern in the Oracle database. See the `functions` command.
`allindexes [[owner_name_pattern.] table_name_pattern]`	Describes the indexes that it finds on the tables that match the input pattern selected from SYS.ALL_OBJECTS. When a pattern is missing, the patterns default to "%". If passthrough is enabled, lists indexes on tables matching the pattern in the Oracle database. See the `indexes` command.

Command	Description
`allpackages [[owner_name_pattern.] table_name_pattern]`	Lists, in a single column, the names of all the PL/SQL packages that match the given pattern selected from SYS.ALL_OBJECTS. When a pattern is missing, the patterns default to "%". If passthrough is enabled, lists PL/SQL packages matching the pattern in the Oracle database. See the `packages` command.
`allprocedures [[owner_name_pattern.] procedure_name_ pattern]`	Lists, in a single column, the names of all the PL/SQL procedures that match the given pattern selected from SYS.ALL_OBJECTS. When a pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists PL/SQL procedures matching the pattern in the Oracle database. See the `procedures` command.
`allsequences [[owner_name_pattern.] table_name_pattern]]`	Lists, in a single column, the names of all the sequences that match the given pattern selected from SYS.ALL_OBJECTS. When a pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists sequences on tables matching the pattern in the Oracle database. See the `sequences` command.
`allsynonyms [[schema_pattern.] object_pattern]]`	Lists, in a single column, the names of all synonyms that match the given pattern. When a pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists synonyms on tables matching the pattern in the Oracle database. See the `synonyms` command.
`alltables [[owner_name_pattern.] table_name_pattern]]`	Lists, in a single column, the names of all the tables that match the given pattern selected from SYS.ALL_OBJECTS. When a pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists tables matching the pattern in the Oracle database. See the `tables` command.
`allviews [[owner_name_pattern.] view_name_pattern]]`	Lists, in a single column, the names of all the views that match the specified pattern selected from SYS.ALL_OBJECTS. When a pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists views matching the pattern in the Oracle database. See the `views` command.
`builtins [builtin_name_ pattern]`	Lists, in a single column, the names of all the TimesTen built-in procedures that match the given pattern. When the pattern is missing, the pattern defaults to "%". See the `procedures` command.
`bye` `exit`	Exits `ttIsql`.

Command	Description
`cachegroups [[cache_group_owner_pattern. cache_group_name_pattern]]`	Reports information on cache groups defined in the currently connected data source, including the state of any terminated databases that contain autorefresh cache groups. If the optional argument is not specified then information on all cache groups in the current data source is reported.
`clearhistory`	Clears the history buffer. Also see `history` and `savehistory`.
`clienttimeout` `[timeeout seconds]`	Sets the client timeout value in seconds for the current connection.
`cachesqlget` `[ASYNCHRONOUS_WRITETHROUGH | INCREMENTAL_AUTOREFRESH] [[cache_group_owner.]cache_group_name] {INSTALL | UNINSTALL} [filename]`	Generates an Oracle SQL*Plus compatible script for the installation or uninstallation of Oracle objects associated with a a read-only cache group, a user managed cache group with incremental autorefresh or an AWT cache group. If `INSTALL` is specified, the Oracle SQL statement to install the Oracle objects is generated. If `UNINSTALL` is specified, the Oracle SQL statement used to remove the Oracle objects is generated. If a cache group is not specified with `UNINSTALL`, a SQL statement to remove all Oracle objects in the AUTOREFRESH user's account is generated. If the optional *`filename`* argument is included, the generated SQL statement is saved to the specified external file. If the external file exists, its contents are destroyed before writing to the file.
`close [connect_id.] command_id]` `closeall`	Closes the prepared command identified by connection name connect_id and command ID command_id. If command_id is not specified, closes the most recent command. If `closeall` is selected, closes all currently open prepared commands.
`cmdcache [[by{sqlcmdid |querytext|owner}] query_subsstring]`	Displays the contents of the TimesTen SQL command cache. Specify the `sqlcmdid`, `querytext` or `owner` column and query substring to search for a specific portion of a SQL query. If no column is specified, searches the `querytext` column. If passthrough is enabled, the command ID is not passed through to the Oracle database.
`commit`	Commits the current transaction (durably if DurableCommits=1 for the connection).
`commitdurable`	Commits the current transaction durably.
`compact`	Compacts the database.

Command	Description
`connect[connection_string | [[DSN][as]connid [adding] [connection_string | DSN][as connid]`	Connects to the database with the specified ODBC *connection_string*. If no password is supplied in this format, `ttIsql` prompts for the password. If no user is given, `ttIsql` attempts to connect using the user name of the current user as indicated by the operating system. If `as` *connid* is specified, you can explicitly name the connection. The `connid` must be only alphanumeric characters, is case sensitive, must start with an alpha character and can only be a maximum of 30 characters in length. The name of `connid` is automatically supplied to the `ConnectionName` general connection attribute. If the connect fails, the current connection is set to a special reserved connection named "none," which is never connected to anything. When `adding` is specified, it refers to creating a new connection to the DSN specified by *DSN* or by the connection string.
`define name [= value]`	Defines a string substitution alias. If no value is provided, `ttIsql` displays the current definition for the specified name. You must `set define on` to enable command substitution. See "Set/show attributes" on page 3-57.
`describe [[owner_pattern.] name_pattern | procedure_ name_pattern | sql_statement | [connect_id.]command_id | *]`	List information on tables, synonyms, views, materialized views, materialized view logs, sequences, PL/SQL functions, PL/SQL procedures, PL/SQL packages, synonyms and TimesTen built-in procedures in that order when the argument is `[owner_pattern.]name_ pattern`. Otherwise lists the specific objects that match the given pattern. Describes the parameters and results columns when the argument is *sql_statement*. If passthrough is set to 3, lists information about the same types of objects in the Oracle database. If `*` is specified, reports the prepared statements for all connections. If the table being described is a materialized view log, the message lists the name of the materialized view for which the table is a log. If the table being described has a materialized view log on it, the message indicates the name of the materialized view log. The command alias is `desc`.
`disconnect` [all]	Disconnects from the database. If `all` is specified, disconnects and closes all connections. When disconnect finishes, the current connection is set to the reserved connection named "none."
`dssize [k|m]`	Prints database size information in KB or MB. The default is KB.
`e:` *msg* `PROMPT` *msg*	Echoes the specified messages, terminated by the end of the line. A semicolon is not required to end the line. Messages are not echoed if verbosity is set to 0.

Command	Description
`exec [connect_id.] command_id] | PLSQLSTMT`	Executes the prepared command command_id on connection connect_id or executes a PL/SQL statement. The *connect_id* optionally names a `ttIsql` connection and command_id is an integer from 1 to 255. If *PLSQLSTMT* is supplied, `ttIsql` prepends the statement with `BEGIN` and appends the statement with `END`, thus allowing the PL/SQL statement to execute. If no argument is supplied, executes the most recent command.
`execandfetch [connect_id.]command_id]`	Executes and fetches all results from prepared command command_id on connection connect_id. If command_id is not specified, executes and fetches all results from the most recent command.
`explain [plan for] {[ Connid.]ttisqlcmdid | sqlcmdid sqlcmdid | sqlstmt |!history}`	Explains the plan for the specified SQL statement, including prepared `ttIsql` statements, specified in the *ttisqlcmdid* argument, or the *sqlcmdid* argument. A digit that is not qualified with the `sqlcmdid` argument, is interpreted as a `ttIsql` prepared statement ID. If passthrough is enabled, the command ID is not passed through to the Oracle database.
`fetchall [connect_id.]command_id]`	Fetches all results from prepared command *command_id* on connection *connect_id*. If *command_id* is not specified, fetches all results from the most recent command. The command must already have been executed using `exec`.
`fetchone [connect_id.]command_id]`	Fetches one result from prepared command *command_id* on connection *connect_id*. If `command_id` is not specified, fetches one result from the most recent command. The command must already have been executed using `exec`.
`free [connect_id.] command_id]`	Frees prepared command *command_id* on connection *connect_id*. If no command is specified, frees the most recent command.
`functions [object_name_pattern]`	Lists, in a single column, the names of PL/SQL functions owned by the current user that match the given pattern. When a name pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists PL/SQL functions matching the pattern in the Oracle database. See the `allfunctions` command.
`globalprocessing statement`	Runs the specified statement with global processing enabled. You cannot specify the statement with the `!` command.

Command	Description
`help [command [command ...]| all |comments | attributes]`	Prints brief or detailed help information for commands. If specific commands are given as arguments then detailed help for each command is printed. If you don't know the exact name of a command, try typing just a few characters that may be part of the command name. `ttIsql` searches and displays help for any commands that include the characters. If `all` is given as an argument then detailed help for all commands is printed. If `comments` is given as an argument then information on using `ttIsql` comments within scripts is printed. If `attributes` is given as an argument then information on the set/show attributes is printed. If no argument is given then brief help information for all commands is printed.
`history` `[-r] [num_commands]`	Lists previously executed commands. The *num_commands* parameter specifies the number of commands to list. If this parameter is omitted, the previous ten commands are listed by default. If the `-r` parameter is specified, commands are listed in reverse order. The history list stores up to 100 of the most recently executed commands. Use the `clearhistory` command to clear the history. See the `savehistory` command.
`host os_command`	Executes an operating system command. The command is executed in the same console as `ttIsql`. This command sets the environment variable TT_CONNSTR in the environment of the process it creates. The value of the variable is the connection string of the current connection.
`indexes [table_name_pattern]`	Describes the indexes that it finds on the tables owned by the current user that match the input pattern. When a name pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists indexes on tables matching the pattern in the Oracle database. See the `allindexes` command.
`monitor`	Formats the contents of the MONITOR table for easy viewing.
`packages [object_name_pattern]`	Lists, in a single column, the names of PL/SQL packages owned by the current user that match the given pattern. When a name pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists PL/SQL packages matching the pattern in the Oracle database. See the `allpackages` command.

Command	Description
`prepare [[connid.]command_id]SQL_Statement`	Prepares the specified SQL statement. If the *command_id* argument is not specified the *command_id* is assigned automatically. The *command_id* argument can take a value between 0 and 255 inclusive. If *connid* is specified, switches to the given connection ID. The *connid* must be only alphanumeric characters and are case insensitive.
`print [variable]`	Prints the value of the specified bind variable or all variables if no variable is specified. If the variable is a refcursor, then the results are fetched and printed.
`procedures [procedure_name_ pattern]`	Lists, in a single column, the names of PL/SQL procedures owned by the current user that match the given pattern. When a name pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists PL/SQL procedures matching the pattern in the Oracle database. See the `builtins` and `allprocedures` commands.
`quit`	Exits `ttIsql`.
`remark msg`	Specifies that the message on the line should be treated as a comment. When `rem` or `remark` is the first word on the line, `ttIsql` reads the line and ignores it.
`repschemes [[scheme_owner_pattern.] scheme_name_pattern]`	Reports information on replication schemes defined in the currently connected data source. This information includes the attributes of all elements associated with the replication schemes. If the optional argument is not specified then information on all replication schemes defined in the current data source is reported.
`retryconnect [0|1]`	Disables(0) or enables(1) the wait for connection retry feature. If the connection retry feature is enabled then connection attempts to a data source that initially fail due to a temporary situation are retried until the connection attempt succeeds. For example, if data source recovery is in progress when attempting to connect, the connection retry feature causes the connect command to continue to attempt a connection until the recovery process is complete. If the optional argument is omitted then the connection retry feature is enabled by default.
`rollback`	Rolls back the current transaction. `AutoCommit` must be off. This command does not stop IMDB Cache operations on Oracle, including passthrough statements, flushing, manual loading, manual refreshing, synchronous writethrough, propagating and transparent loading.

Command	Description

`run` *`filename`* `[`*`arguments`*`]|`
`start` *`filename`* `[`*`arguments...`*`]|`
`@@` *`filename`* `[`*`arguments...`*`]|`
`@` *`filename`* `[`*`arguments...`*`]`
	Reads and executes SQL commands from filename. The run command can be nested up to five levels.
	The `@@` command is identical to the `@` command only if the file is specified with an absolute path.
	When `@` is used with a relative path, the path is relative to the startup directory of `ttIsql`. When `@@` is used, it is relative to the currently running input file. Therefore `@@` is useful when used in a script that must call other scripts. It does not matter what directory the invoker of `ttIsql` is in when the script is run.
	See "Example parameters of command string substitution" on page 3-64 for a description of *`arguments`*.

`savehistory` `[-a | -f]` *`outputfile`*	Writes the history buffer to the specified *`output`* file. Only command, no parameter values are saved in the output file. Therefore, a script may not be able to replay the history from the output file. If the *`output`* file exists, you must specify either the `-a` or `-f` option. If `-a` is specified, the history is appended to the specified *`output`* file. If `-f` is specified, the history overwrites the contents of the specified *`output`* file. See the `clearhistory` and `history` commands.
`sequences` `[`*`sequence_name_pattern`*`]`	Lists, in a single column, the names of sequences owned by the current user that match the given pattern. When a name pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists sequences on tables matching the pattern in the Oracle database. See the `allsequences` command.
`set` *`attribute`* `[`*`value`*`]`	Sets the specified attribute to the specified value. If no value is specified, displays the current value of the specified attribute. For a description of accepted attributes, see "Set/show attributes" on page 3-57.
`setjoinorder` *`tblNames`* `[...]`	Specifies the join order for the optimizer. `AutoCommit` must be off.
`setuseindex` *`index_name`*`,`*`correlation_name`*`,` `{0 | 1} [;...]`	Sets the index hint for the query optimizer.
`show {all |` *`attribute`*`}`	Displays the value for the specified database attribute or displays all the attributes. For a description of accepted attributes, see "Set/show attributes" on page 3-57.
`showjoinorder {0 | 1}`	Enables or disables the storing of join orders. 0 - Disables the storing of join orders 1 - Enables the storing of join orders. Call the `ttoptshowjoinorder` built-in procedure explicitly to display the join order after `SELECT`, `UPDATE`, `DELETE` or `MERGE` SQL statements.

Command	Description
`sleep [n]`	Suspends execution for *n* seconds. If *n* is not specified then execution is suspended for 1 second.
`spool filename {[option] | OFF}`	Writes a copy of the terminal output to the file filename. If you do not provide an extension to the filename, the filename has the extension `.lst`. The available options include ■ **CREATE** - Creates a new file. ■ **APPEND** - Appends output to an existing file. ■ **REPLACE** (default) - Overwrites an existing file. When you specify the value OFF, the spooling behavior is terminated and the output file is closed. If you specify a spool command while one is running, the active spool is closed and a new files is opened.
`sqlcolumns [owner_name_pattern.] table_name_pattern`	Prints results of an ODBC call to `SQLColumns`.
`sqlgetinfo infotype`	Prints results of an ODBC call to `SQLGetInfo`.
`sqlstatistics [[owner_name_pattern.] table_name_pattern]`	Prints results of an ODBC call to `SQLStatistics`.
`sqltables [[owner_name_pattern.] table_name_pattern]`	Prints results of a call to SQLTables. The pattern is a string containing an underscore (_) to match any single character or a percent sign (%) to match zero or more characters.
`statsclear [[owner_name.]table_name]`	Clears statistics for specified table (or all tables if no table is specified).
`statsestimate [[owner_name.]table_name] {n rows | p percent}`	Estimates statistics for specified table (or all tables if no table is specified).
`statsupdate [[owner_name_pattern.] table_name_pattern]`	Updates statistics for specified table (or all tables if no table is specified).
`synonyms [[schema_pattern.] object_pattern]]`	Lists, in a single column, the names of synonyms owned by the current user that match the given pattern. When a name pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists synonyms on tables matching the pattern in the Oracle database. See the `allsynonyms` command.
`tables [table_name_pattern]]`	Lists, in a single column, the names of tables owned by the current user that match the given pattern. When a name pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists tables matching the pattern in the Oracle database. See the `alltables` command.
`tryglobalprocessing [0|1]`	Disables (`0`) or enables (`1`) the global processing of eligible queries in the cache grid. If the optional argument is omitted, global processing is enabled. Use `show optprofile` to query `GlobalProcessing`. `AutoCommit` must be off.
`undefine name`	Undefines a string substitution alias.

Command	Description
`unsetjoinorder`	Clears join order advice to optimizer. `AutoCommit` must be off.
`unsetuseindex`	Clears the index hint for the query optimizer.
`use [conn_id]`	Displays the list of current connections and their IDs. If connid is specified, switches to the given connection ID. To use the name of the first connection, you can specify `con0` for the *`conn_id`*, rather than specifying the full original connection name. You cannot explicitly name a connection `con0`. If the first connection is disconnected, `con0` refers to the connection `none`. If `use` fails to locate the connection id, the current connection is set to the reserved connection named "none." See the `connect` command.
`variable [variable [type]]`	Declares a bind variable that can be referenced in a statement, or displays the definition of the variable.
`version`	Reports version information.
`views [table_name_pattern]`	Lists, in a single column, the names of views owned by the current user that match the given pattern. When a name pattern is missing, the pattern defaults to "%". If passthrough is enabled, lists views matching the pattern in the Oracle database. See the `allviews` command.
`xlabookmarkdelete id`	Deletes a persistent XLA bookmark. If a bookmark to delete is not specified then the status of all current XLA bookmarks is reported. See "XLA Reference" in *Oracle TimesTen In-Memory Database C Developer's Guide*. Requires `ADMIN` privilege or object ownership.

Set/show attributes

Also see the list of `ttIsql` "Commands" on page 3-48. Some commands appear here as attributes of the `set` command. In that case, they can be used with or without the `set` command.

Boolean attributes can accept the values "`ON`" and "`OFF`" in place of "`1`" and "`0`".

`ttIsql` set supports these attributes:

Attribute	Description
`all`	With show command only. Displays the setting of all the `ttIsql` commands.
`autocommit [1|0]`	Turns `AutoCommit` off and on. If no argument is given, displays the current setting.

Attribute	Description
`columnlabels [0 | 1]`	Turns the `columnlabels` feature off (0) or on (1). If no argument is specified, the current value of `columnlabels` is displayed. The initial value of `columnlabels` is off (0) after connecting to a data source. When the value is on (1), the column names are displayed before the SQL results. You can also enable this attribute without specifying the set command.
`connstr`	Prints the connection string returned from the driver from the `SQLDriverConnect` call. This is the same string printed when `ttIsql` successfully connects to a database.
`define [&|c|on|off]`	Sets the character used to prefix substitution variables to `c`. `ON` or `OFF` controls whether `ttIsql` scans commands for substitution variables and replaces them with their values. `ON` changes the value of `c` back to the default "&". (It does not change it to the most recently used character.) Default value for `ttIsql` is `OFF` (no variable substitution). See "Example parameters using "variable" and "print"" on page 3-68 for an explanation of the default.
`dynamicloadenable [on|off]`	Enables or disables dynamic load of Oracle data to a TimesTen dynamic cache group. By default, dynamic load of Oracle data is enabled.
`echo [on | off]`	With the `set` command, prints the commands listed in a `run`, `@` or `@@` script to the terminal as they are executed. If off, the output of the commands is printed but the commands themselves are not printed.
`editline [0 | 1]`	Turns the editline function off and on. By default, editline is on. If editline is turned off, the backspace character deletes full characters, but the rest of editline capabilities are unavailable.
`err | error |errors [.`*`objecttype`*`[`*`schema`*`.] `*`name`*`]`	Shows errors command display error information about the given PL/SQL object. If no object type or object name is supplied, `ttIsql` assumes the PL/SQL object that you last attempted to create and retrieves the errors for that object. If no errors associated with the given object are found, or there was no previous PL/SQL DDL, then `ttIsql` displays "No errors."
`feedback [on | off] `*`rows`*	Controls the display of status messages after statement execution. When `rows` is specified, if the statement affected more than the specified number of rows, then the feedback indicates the number of affected rows. If the number of rows affected is less than the specified threshold, the number of rows is not printed. Feedback is not provided for tables, views, sequences, materialized views or indexes. It is available for PL/SQL objects.

Attribute	Description
`isolation [{READ_COMMITTED | 1}| {SERIALIZABLE | 0}]`	Sets isolation level. If no argument is supplied, displays the current value. You can also enable this attribute without specifying the set command.
`multipleconnections [1 | ON] mc [1 | ON]`	Reports or enables handling of multiple connections.By default, `ttIsql` allows the user to have one open connection at a time. If the argument `1` or `ON` is specified the prompt is changed to include the current connection and all multipleconnection features are enabled. If no value is supplied, the command displays the value of the `multipleconnections` setting. You can also enable this attribute without specifying the set command.
`ncharencoding` [*encoding*]	Specifies the character encoding method for `NCHAR` output. Valid values are `LOCALE` or `ASCII`. `LOCALE` sets the output format to the locale-based setting. If no value is specified, TimesTen uses the system's native language characters. You can also enable this attribute without specifying the set command.
`optfirstrow` [1|0]	Enables or disables First Row Optimization. If the optional argument is omitted, First Row Optimization is enabled. You can also enable this attribute without specifying the set command.
`optprofile`	Prints the current optimizer flag settings and join order. This attribute cannot be used with the set command.

Attribute	Description
`passthrough [0|1|2|3|4|5]`	Sets the IMDB Cache passthrough level for the current transaction. Because `AutoCommit` must be off to execute this command, `ttIsql` temporarily turns off `AutoCommit` when setting the passthrough level. **0** - SQL statements are executed only against TimesTen. **1** - Statements other than `INSERT`, `DELETE` or `UPDATE` and DDL are passed through if they generate a syntax error in TimesTen or if one or more tables referenced within the statement are not in TimesTen. All `INSERT`, `DELETE` and `UPDATE` statements are passed through if the target table cannot be found in TimesTen. DDL statements are not passed through. **2** - Same as 1, plus any `INSERT`, `UPDATE` and `DELETE` statement performed on `READONLY` cache group tables is passed through. **3** - All SQL statements, except `COMMIT` and `ROLLBACK`, and TimesTen built-in procedures that set or get optimizer flags are passed through. `COMMIT` and `ROLLBACK` are executed on both TimesTen and Oracle. **4** - All `SELECT` statements on global cache groups tables that cannot use transparent load are executed on Oracle. **5** - All `SELECT` statements on global cache groups tables that cannot use transparent load are executed on Oracle. The `SELECT` statement is not executed until after all committed changes to the global cache group are propagated to Oracle. If no optional argument is supplied, the current setting is displayed. After the transaction, the passthrough value is reset to the value defined in the connection string or in the DSN or the default setting if no value was supplied to either. You can also enable this attribute without specifying the set command. **Note:** Some Oracle objects may not be described by `ttIsql`.
`prefetchcount` [*`prefetch_count_size`*]	Sets the prefetch count size for the current connection. If the optional argument is omitted, the current prefetch count size is reported. Setting the prefetch count size can improve result set fetch performance. The *`prefetch_count_size`* argument can take an integer value between 0 and 128 inclusive. You can also enable this attribute without specifying the set command.
`prompt [string]`	Replaces the `Command>` prompt with the specified string. To specify a prompt with spaces, you must quote the string. The leading and trailing quotes are removed. A prompt can have a string format specifier (`%c`) embedded. The `%c` is expanded with the name of the current connection.
`querythreshold` [*`seconds`*]	With the `show` command, displays the value of the Query Threshold first connection attribute. With the `set` command, modifies the value of the QueryThreshold first connection attribute that was set in the connection string or `odbc.ini` file. Specify a value in seconds that indicates the number of seconds that a query can execute before TimesTen writes a warning to the support log or throws an SNMP trap.

Attribute	Description
`rowdelimiters [0|off] | [ {1|on} [begin end]]`	Controls the row delimiters in result sets. When on, user queries have the row delimited with '< ' and ' >' unless *begin* and *end* are specified. Not all result sets are affected by this control. The default is on.
`serveroutput [on | off]`	With the `set` command set to on, after each executed SQL statement, displays any available output. This output is available for debugging I/O purposes, if the PL/SQL `DBMS_OUTPUT` package is set to store the output so that it can be retrieved using this command. The default is off, (no server output is displayed) as performance may be slower when using this command. If you set serveroutput to on, TimesTen uses an unlimited buffer size. `DBMS_OUTPUT.ENABLE` is per connection, therefore `set serveroutput on` affects the current connection only.
`showplan [0 | 1]`	Enables (`1`) or disables (`0`) the display of plans for selects/updates/deletes in this transaction. If the argument is omitted, the display of plans is enabled. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`sqlquerytimeout [seconds]`	Specifies the number of seconds to wait for a SQL statement to execute before returning to the application for all subsequent calls. If no time or 0 seconds is specified, displays the current timeout value. The value of *`seconds`* must be equal to or greater than 0.This attribute does not stop IMDB Cache operations on Oracle, including passthrough statements, flushing, manual loading, manual refreshing, synchronous writethrough, propagating, and transparent loading. You can also enable this attribute without specifying the set command.
`timing [1|0]`	Enables or disables printing of query timing. You can also enable this attribute without specifying the set command.
`tryhash [1|0]`	Enables or disables use of hash indexes by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`trymaterialize [1|0]`	Enables or disables materialization by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`trymergejoin [1|0]`	Enables or disables use of merge joins by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.

Attribute	Description
`trynestedloopjoin [1|0]`	Enables or disables use of nested loop joins by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`tryrowid [1|0]`	Enables or disables rowID scan hint.
`tryrowlocks [1|0]`	Enables or disables use of row-level locking by the optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`tryserial [1|0]`	Enables or disables use of serial scans by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`trytmphash [1|0]`	Enables or disables use of temporary hashes by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`trytbllocks [1|0]`	Enables or disables use of table-level locking by the optimizer. `AutoCommit` must be off. You can also set this attribute without specifying the set command.
`trytmptable [1|0]`	Enables or disables use of temporary tables by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`trytmpttree [1|0]`	Enables or disables use of temporary range indexes by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`tryttree [1|0]`	Enables or disables use of range indexes by optimizer. `AutoCommit` must be off. You can also enable this attribute without specifying the set command.
`verbosity` [*`level`*]	Changes the verbosity level. The verbosity level argument can be an integer value of 0,1,2,3 or 4. If the optional argument is omitted then the current verbosity level is reported. You can also enable this attribute without specifying the set command.
`vertical` [{0 | off} | {1 | on} | *`statement`*]	Sets or displays the current value of the vertical setting. The default value is `0` (`off`). If statement is supplied, the command temporarily turns vertical on for the given statement. This form is only useful when the vertical flag is off. The `vertical` setting controls the display format of result sets. When set, the result sets are displayed in a vertical format where each column is on a separate line and is displayed with a column label. You can also enable this attribute without specifying the set command.

Comment syntax

The types of comment markers are:

```
# [comment_text]
-- [comment_text]
/* [comment_text] */
```

The C-style comments (`/* [comment_text] */`) can span multiple lines.

The comments delimited by the

#

and the -

–

characters should not span multiple lines. If a comment marker is encountered while processing a line, then the remainder of the line is ignored.

'`--`' at the beginning of a line is considered a SQL comment. The line is considered a comment and no part of the line is included in the processing of the SQL statement. A line that begins with '`--+`' is interpreted as a segment of a SQL statement.

The comment markers can work in the middle of a line.

Example:

```
monitor; /*this is a comment after a ttIsql command*/
```

Command history

ttIsql implements a `csh`-like command history.

Command Usage: `history [-r] [num_commands]`

Description: Lists previously executed commands. The *`num_commands`* parameter specifies the number of commands to list. If the `-r` parameter is specified, commands are listed in reverse order.

Command Usage: `! [command_id|command_string| !]`

Description: Executes a command in the history list. If a *`command_id`* argument is specified, the command in the history list associated with this ID is executed again. If the *`command_string`* argument is specified, the most recent command in the history list that begins with command_string is executed again. If the `!` argument is specified then the most recently executed command is executed again.

Example: `"!!;"` -or- `"!10;"` -or- `"!con;"`

Also see the `clearhistory`, `history`, `savehistory` commands.

Command shortcuts

By default, `ttIsql` supports keystroke shortcuts when entering commands. To turn this feature off, use:

```
Command> set editline=0;
```

The bindings available are:

Keystroke	Action
Left Arrow	Moves the insertion point left (back).

Keystroke	Action
Right Arrow	Moves the insertion point right (forward).
Up Arrow	Scroll to the command before the one being displayed. Places the cursor at the end of the line.
Up Arrow <RETURN>	Scrolls to the PL/SQL block before the one being displayed.
Down Arrow	Scrolls to a more recent command history item and puts the cursor at the end of the line.
Down Arrow <RETURN>	Scrolls to the next PL/SQL block after the one being displayed.
Ctrl-A	Moves the insertion point to the beginning of the line.
Ctrl-E	Moves the insertion point to the end of the line.
Ctrl-K	"Kill" - Saves and erases the characters on the command line from the current position to the end of the line.
Ctrl-Y	"Yank"- Restores the characters previously saved and inserts them at the current insertion point.
Ctrl-F	Forward character - move forward one character. (See Right Arrow.)
Ctrl-B	Backward character - moved back one character. (See Left Arrow.)
Ctrl-P	Previous history. (See Up Arrow.)
Ctrl-N	Next history. (See Down Arrow.)

Parameters

With dynamic parameters, you are prompted for input for each parameter on a separate line. Values for parameters are specified the same way literals are specified in SQL.

SQL_TIMESTAMP columns can be added using dynamic parameters. (For example, values like `'1998-09-08 12:1212'`).

Parameter values must be terminated with a semicolon character.

The possible types of values that can be entered are:

- Numeric literals. Example: `1234.5`
- Time, date or timestamp literals within single quotation marks. Examples:

  ```
  '12:30:00''2000-10-29''2000-10-29 12:30:00''2000-10-29 12:30:00.123456'
  ```

- Unicode string literals within single quotation marks preceded by `'N'`. Example: `N'abc'`
- A NULL value. Example: `NULL`
- The '*' character which indicates that the parameter input process should be stopped. Example: `*`
- The '?' character prints the parameter input help information. Example: `?`

Example parameters of command string substitution

```
Command> select * from dual where :a > 100 and :b < 100;
Type '?' for help on entering parameter values.
Type '*' to end prompting and abort the command.
Type '-' to leave the parameter unbound.
Type '/;' to leave the remaining parameters unbound and execute the command.
```

```
Enter Parameter 1 'A' (NUMBER) > 110
Enter Parameter 2 'B' (NUMBER) > 99
< X >
1 row found.
Command> var a number;
Command> exec :a := 110;

PL/SQL procedure successfully completed.

Command> print a
A                    : 110
Command> var b number;
Command> exec :b := 99;

PL/SQL procedure successfully completed.

Command> select * from dual where :a > 100 and :b < 100;
< X >
1 row found.
Command> print
A                    : 110
B                    : 99
Command> select * from dual where :a > 100 and :b < 100 and :c > 0;
Enter Parameter 3 'C' (NUMBER) > 1
< X >
1 row found.
Command>
```

Default options

You can set the default command-line options by exporting an environment variable called `TTISQL`. The value of the `TTISQL` environment variable is a string with the same syntax requirements as the `TTISQL` command line. If the same option is present in the `TTISQL` environment variable and the command line then the command line version always takes precedence.

Examples

Execute commands from `ttIsql.inp`.

```
ttIsql -f ttIsql.inp
```

Enable all output. Connect to DSN RunData and create the database if it does not exist.

```
ttIsql -v 4 -connStr "DSN=RunData;AutoCreate=1"
```

Print the interactive commands.

```
ttIsql -helpcmds
```

Print the full help text.

```
ttIsql -helpfull
```

Display the setting for all `ttIsql` attributes:

```
Command> show all;
Connection independent attribute values:

autoprint = 0 (OFF)
columnlabels = 0 (OFF)
```

```
define = 0 (OFF)
echo 1 (ON)
FEEDBACK ON
multipleconnections =0 (OFF)
ncharencoding = LOCALE (US7ASCII)
prompt = 'COMMAND>'
timing = 0 (OFF)
verbosity = 2
vertrical = 0 (OFF)

Connection specific attribute values:

autocommit = 1 (ON)
Client timeout = 0
Connection String DSN=repdb1_1121;UID=timesten; DataStore=/DS/repdb1_1121;
 DatabaseCharacterSet=AL32UTF8; ConnectionCharacterSet=US7ASCII;
 DRIVER=/opt/TimesTen/tt1121/lib/libtten.so; PermSize=20;TempSize=20;TypeMode=1;
No errors.
isolation = READ_COMMITTED
Prefetch count = 5
Query threshold = 0 seconds (no threshold)
Query timeout = 0 seconds (no timeout)
serveroutput OFF

Current Optimizer Settings:
    Scan: 1
    Hash: 1
    Ttree: 1
    TmpHash: 1
    TmpTable: 1
    NestedLoop: 1
    MergeJoin: 1
    GenPlan: 0
    TblLock: 1
    RowLock: 1
    Rowid: 1
    FirstRow: 1
    IndexedOr: 1
    PassThrough: 0
    BranchAndBound: 1
    ForceCompile: 0
    CrViewSemCheck: 1
    ShowJoinOrder: 0
    CrViewSemCheck: 1
    TransparentLoad: 0
    UserBoyerMooreStringSearch: 0
    DynamicLoadEnable: 1
    DynamicLoadErrorMode: 0
    NoRemRowIdOpt: 0

Current Join Order:
    <>

Command
```

Prepare and exec an SQL statement.

```
ttIsql (c) 1996-2009, TimesTen, Inc. All rights reserved.
ttIsql -connStr "DSN=RunData"
Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
(Default setting AutoCommit=1)
Command> prepare 1 SELECT * FROM my_table;
Command> exec 1;
Command> fetchall;
```

Example `vertical` command:

```
Command> call ttlogholds;
< 0, 265352, Checkpoint , DS.ds0 >
< 0, 265408, Checkpoint , DS.ds1 >
2 rows found.
Command> vertical call ttlogholds;

 HOLDLFN:       0

 HOLDLFO:       265352
 TYPE:          Checkpoint
 DESCRIPTION:   DS.ds0
 HOLDLFN:       0

 HOLDLFO:       265408
 TYPE:          Checkpoint
 DESCRIPTION:   DS.ds1
 2 rows found.

Command>
```

To create a new user, use single quotes around the password name for an internal user:

```
ttIsql -connStr "DSN=RunData"
ttIsql (c) 1996-2000, TimesTen, Inc. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
(Default setting AutoCommit=1)
Command> CREATE USER terry IDENDTIFIED BY `secret';
```

To delete the XLA bookmark `mybookmark`, use:

```
ttIsql -connStr "DSN=RunData"
ttIsql (c) 1996-2000, TimesTen, Inc. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql. (Default setting
AutoCommit=1)
Command> xlabookmarkdelete;
XLA Bookmark: mybookmark
 Read Log File:  0
 Read Offset:    268288
 Purge Log File: 0
 Purge Offset:   268288
 PID:            2004
 In Use:         No
1 bookmark found.

Command> xlabookmarkdelete mybookmark;

Command> xlabookmarkdelete;

0 bookmarks found.
```

Example parameters using "variable" and "print"

Substitution in `ttIsql` is modeled after substitution in SQL*Plus. The substitution feature is enabled by 'set define on' or 'set define <substitution_char>'. The substitution character when the user specifies 'on' is '&'. It is disabled with 'set define off'.

By default, substitution is off. The default is off because the '&' choice for substitution character conflicts with TimesTen's use of ampersand as the BIT AND operator.

When enabled, the alphanumeric identifier following the substitution character is replace by the value assigned to that identifier. When disabled, the expansion is not performed.

New definitions can be defined even when substitution is off. You can use the "define" command to list the definitions `ttIsql` predefines.

```
Command> show define
define = 0 (OFF)
Command> define
DEFINE            _PID = "9042" (CHAR)
DEFINE      _O_VERSION = "TimesTen Release 11.2.1.0.0" (CHAR)
Command> select '&_O_VERSION' from dual;
< &_O_VERSION >
1 row found.
Command> set define on
Command> SELECT '&_O_VERSION' FROM DUAL;
< TimesTen Release 11.2.1.0.0 >
1 row found.
```

If the value is not defined, `ttIsql` prompts you for the value.

When prompting and only one substitution character is used before the identifier, the identifier is defined only for the life of the one statement.

If two substitution characters are used and the value is prompted, it acts as if you have explicitly defined the identifier.

```
Command> SELECT '&a' FROM DUAL;
Enter value for a> hi
< hi >
1 row found.
Command> define a
symbol a is UNDEFINED
The command failed.
Command> SELECT '&&a' FROM DUAL;
Enter value for a> hi there
< hi there >
1 row found.
Command> define a
DEFINE               a = "hi there" (CHAR)
```

Additional definitions are created with the define command:

```
Command> define tblname = sys.dual
Command> define tblname
DEFINE         tblname = "sys.dual" (CHAR)
Command> select * from &tblname;
< X >
1 row found.
```

Arguments to the run command are automatically defined to &1, &2, ... when you add them to the 'run' or '@' (and '@@') commands:

Given this script:

```
CREATE TABLE &1 ( a INT PRIMARY KEY, b CHAR(10) );
INSERT INTO &1 VALUES (1, '&2');
INSERT INTO &1 VALUES (2, '&3');SELECT * FROM &1;
```

Use the script:

```
Command> SET DEFINE ON
Command> @POPULATE mytable Joe Bob;

CREATE TABLE &1 ( a INT PRIMARY KEY, b CHAR(10) );
INSERT INTO &1 VALUES (1, '&2');
1 row inserted.

INSERT INTO &1 VALUES (2, '&3');
1 row inserted.

SELECT * FROM &1;
< 1, Joe        >
< 2, Bob        >
2 rows found.
Command>
```

This example uses the `variable` command. It deletes an employee from the `employee` table. Declare `empid` and `name` as variables with the same data types as `employee_id` and `last_name`. Delete the row, returning `employee_id` and `last_name` into the variables. Verify that the correct row was deleted.

```
Command> VARIABLE empid NUMBER(6) NOT NULL;
Command> VARIABLE name VARCHAR2(25) INLINE NOT NULL;
Command> DELETE FROM employees WHERE last_name='Ernst'
       > RETURNING employee_id, last_name INTO :empid,:name;
1 row deleted.
Command> PRINT empid name;
EMPID                : 104
NAME                 : Ernst
```

Notes

Multiple `ttIsql` commands are allowed per line separated by semicolons.

The `ttIsql` utility command line accepts multiline PL/SQL statements, such as anonymous blocks, that are terminated with the "/" on it's own line. For example:

```
Command> set serveroutput on
Command> BEGIN
> dbms_ouput.put_line ('Hi There');
> END;
>/
Hi There

PL/SQL block successfully executed.

Command>
```

For UTF-8, NCHAR values are converted to UTF-8 encoding and then output.

For ASCII, those NCHAR values that correspond to ASCII characters are output as ASCII. For those NCHAR values outside of the ASCII range, the escaped Unicode format is used. For example:

```
U+3042 HIRAGANA LETTER A
```

is output as

```
Command> SELECT c1 FROM t1;
< a\u3042 >
```

NCHAR parameters must be entered as ASCII N-quoted literals:

```
Command> prepare SELECT * FROM t1 WHERE c1 = ?;
Command> exec;
```

Type `'?;'` for help on entering parameter values. Type `'*;'` to abort the parameter entry process.

```
Enter Parameter 1> N'XY';
```

WINDOWS

On Windows, this utility is supported for all TimesTen Data Manager and Client DSNs.

ttMigrate

Description

Performs one of these operations:

- Saves a migrate object from a TimesTen database into a binary datafile.
- Restores the migrate object from the binary datafile into a TimesTen database.
- Examines the contents of a binary datafile created by this utility.

Migrated objects include:

- Tables
- Cache group definitions
- Views and materialized views
- Materialized view log definitions
- Sequences
- Replication schemes

The `ttMigrate` utility is used when upgrading major release versions of TimesTen, since major revisions are not compatible. For an example, see the *Oracle TimesTen In-Memory Database Installation Guide*.

When you migrate a database into Release 11.2.1 from a previous release, users and user privileges are not migrated. When you migrate a database between releases of Release 11.2.1 or into a release later than Release 11.2.1, users and user privileges are migrated.

Binary files produced by this utility are platform-dependent. For example a binary file produced on Windows must be restored on Windows. Use the `ttBulkCp` utility to copy data between platforms.

The `ttMigrate` utility can be used to copy data between bit-levels within the same architecture. For example, it can be used to move data from a 32-bit Solaris system to a 64-bit Solaris system. The `-noRepUpgrade` option must be used when changing bit-levels and the database should not be involved in a replication scheme, in this case.

By default, `ttMigrate` restores the database using one thread. During restoration, you can specify the `-numThreads` option to restore the data files using multiple threads, thus potentially improving performance.

UNIX

On UNIX, this utility is supported for TimesTen Data Manager DSNs. For TimesTen Client DSNs, use the utility `ttMigrateCS`.

Required privilege

This utility requires various privileges depending on the options specified. In general, a user must be the instance administrator or have the ADMIN privilege to use this utility.

Using the `-r` option requires the instance administrator privilege, as it generally creates a database. If the database has been created at the time this option is used, it requires CREATE ANY TABLE, CREATE ANY SEQUENCE, CREATE ANY VIEW, CREATE ANY MATERIALIZED VIEW, CREATE ANY CACHE GROUP, CREATE ANY INDEX privileges and ADMIN if autocreation of users is necessary. If the

database is involved in replication or IMDB Cache, then CACHE_MANAGER is also required.

Using the `-c` option to capture an entire database requires the ADMIN privilege.

Using the `-c` option to capture a subset of the database objects (tables, views, materialized views, cache groups, sequences) requires SELECT ANY TABLE and SELECT ANY SEQUENCE privileges.

Syntax

```
ttMigrate {-h | -help | -?}

ttMigrate {-V | -version}
```

To create or append a binary datafile, use:

```
ttMigrate {-a | -c} [-v verbosity] [-nf] [-nr] [-fixNaN]
[-saveAsCharset charset] [-repUpgrade | -noRepUpgrade
[-convertTypesToOra | -convertTypestoTT]]
{-connStr connection_string | DSN} dataFile [objectOwner.]objectName
```

To restore a database from a binary datafile created by this utility, use:

```
ttMigrate -r -noRepUpgrade  [-numThreads n] [-inline rule] [-v verbosity]
 [-fixNaN] [-nf] [-nr] [-C chkPtFreq] [-rename oldOwner:newOwner] [-localhost
 hostName] [-n noCharsetConversion] [-cacheUid uid [-cachePwd pwd]]
 [-convertCGtypes]  [-autorefreshPaused] [-convertTypesToOra | -convertTypesToTT]
 [-noAutoCreateUsers] -connStr connection_string | DSN} dataFile
 [[objectOwner.]objectName]
```

or

```
ttMigrate -r -repUpgrade  [-numThreads n] [-vverbosity] [-fixNaN] [-nf] [-nr]
[-C chkPtFreq] [-rename oldOwner:newOwner] [-localhost hostName]
[-noCharsetConversion] [-cacheUid uid [-cachePwd pwd]] [-convertCGtypes]
[-updateStats | -estimateStats percentRows] {-connStr connection_string | DSN}
dataFile [[objectOwner.]objectName]
```

To list or display the contents of a binary datafile created by this utility, use:

```
ttMigrate {-l | -L | -d | -D} dataFile [[objectowner.]name ...]
```

Options

> **Note:** The append (-a) or create (-c) modes, the list (-l/-L) or describe (-d/-D) modes and the restore (-r) modes are exclusive of each other. You cannot specify any of these options on the same line as any other of these options.

`ttMigrate` has the options:.

Option	Description
`-a`	Selects append mode: Appends data to a pre-existing binary data file, that was originally created using `ttMigrate -c`. See "Create mode (-c) and Append mode (-a)" on page 3-77 for more details.
`-c`	Create mode: Creates an original binary datafile. See "Create mode (-c) and Append mode (-a)" on page 3-77 for more details.

Option	Description
`-cacheUid`	The cache administration user ID to use when restoring asynchronous writethrough cache groups and cache groups with the AUTOREFRESH attribute.
`-cachePwd`	The cache administration password to use when restoring AUTOREFRESH and asynchronous writethrough cache groups and cache groups with the AUTOREFRESH attribute. If the cache administration user ID is provided on the command line but the cache administration password is not, then `ttMigrate` prompts for the password.
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.

`-convertTypesToOra`
 	
`-convertTypesToTT`
	Converts TimesTen data types to Oracle data types or Oracle data types to TimesTen data types. These options require the`-noRepUpgrade` option.
	
	In TimesTen 11.2.1 the default type mode is ORACLE type mode. The `-convertTypesToOra` is useful when migrating older databases into TimesTen 11.2.1.
	
	The `-convertTypesToTT` option is useful to allow backward migration into a release that does not support Oracle types.
	
	These options apply to all table types except materialized views. Table types include: regular, cached, global and temporary tables.
	
	See also `-convertCGTypes`, "TimesTen to Oracle data type conversions" on page 3-80 and "Oracle to TimesTen data type conversions" on page 3-81.

`-d`	Selects Describe mode. Displays a short description of the objects in the datafile. See "Describe mode (-d)" on page 3-79 for more details.
`-D`	Selects Long-describe mode. Displays a full description of the objects in the datafile. See "Long-describe mode (-D)" on page 3-80 for more details.
`dataFile`	The path name of the datafile to which migrate objects are to be saved or from which migrate objects are to be restored.
`DSN`	Specifies an ODBC data source name of the database to be migrated.
`-estimateStats` *`percent`*	Specifies that `ttMigrate` should estimate statistics on restored tables and materialized views for the specified percentage of rows. Legal values for percentRows are 0 to 100, inclusive. This option is ignored when the `-c or -a` options are given. If you specify both `-estimateStats` and `-updateStats`, statistics on restored tables are updated, not estimated. Use of this flag may improve the performance of materialized view restoration and may also improve the performance of queries on the restored tables and views.
`-fixNaN`	Converts all NaN, Inf and -Inf values found in migrate objects to 0.0. This is useful for migrating data into releases of TimesTen that do not support the NaN, Inf and -Inf values.

`-h`
 	
`-help`
 	
-?
	Prints a usage message and exits.

Option	Description
`-inline` *`rule`*	Indicates the rule to be used for converting variable-length columns to INLINE in restore mode. The value for rule is one of: `preserve` - `ttMigrate` preserves the original `INLINE` attribute of each column. This is the default, and it is required if`-repUpgrade` is used. `dsDefault` - `ttMigrate` uses the database's default rule for setting the `INLINE` attribute of restored columns. *`maxlen`* - `ttMigrate` restores as `INLINE` all variable-length columns with length <= *`maxlen`* and restores as `NOT INLINE` all variable-length columns with length greater than maxlen. If *`maxlen`* is 0 then all columns are restored as `NOT INLINE`.When this option is used during migration, the database should not be replicated with databases from a TimesTen version that does not support `INLINE` columns. `INLINE` variable-length columns cannot successfully be replicated to `NOT INLINE` columns.
`-l`	Selects List mode. Lists the names of database objects in the specified datafile. See "List mode (-l) and Long-list mode (-L)" on page 3-79 for more details.
`-L`	Selects Long-list mode. Lists the names of database objects in the specified datafile and other details about the database objects. See "List mode (-l) and Long-list mode (-L)" on page 3-79 for more details.
`-r`	Selects Restore mode. Restores a database from a binary datafile created by this utility. See "Restore mode (-r)" on page 3-78 for more details.
`name`	The name of the database object(s) to be saved or restored.
`-nf`	Specifies that `ttMigrate` should not save or restore foreign key information when saving or restoring ordinary (non-cached) tables.
`-nr`	Specifies that `ttMigrate` should not save or restore table rows when saving or restoring ordinary (non-cached) tables.
`-noAutoCreateUsers`	Specifies that `ttMigrate` should not create users. By default, TimesTen creates "disabled" users when migrating tables from releases earlier than 11.2.1. TimesTen creates users but does not assign any privileges to these users. You must explicitly assign privileges, including `CREATE SESSION`, to these users after they are created.
`-noRepUpgrade`	Do not ensure that saved/restored tables are compatible for replication with the original tables. Use of this option may cause the restored tables to be slightly more compact and slightly faster to access than otherwise. This option is ignored when the `-a` option is given. This option should be used with care, however, as replication between the original tables and the restored tables may not work properly. INLINE variable-length columns cannot successfully be replicated to NOT INLINE columns. If you specify this option, replication is only possible if you have specified RELAXED for your replication scheme.

Option	Description
`-numThreads` *n*	Specifies the number of threads to use while restoring a database files. If unspecified, `ttMigrate` uses one thread to restore objects from the data file. Valid values are 1 through 32.
owner	The owner of a migrate object.
`-rename` *oldOwner*:*newOwner*	Renames user *oldUser* to be *newUser*. If *newUser* exists in the database, the operation returns an error and stops. Also restores all database objects previously owned by *oldUser* to be owned by *newUser*. Also grants *newUser* all privileges previously granted to *oldUser*. You cannot rename the TimesTen system users (`TTREP`, `SYS`, `SYSTEM`, `GRID`). This option is restricted to only work in databases that have PL/SQL disabled (`PLSQL=0`). For databases with PL/SQL enabled, consider using synonyms to achieve comparable behavior. This option should be used with some caution. When using the `-rename` option, be aware that: ■ If the old user name does not exist, a warning is printed at the end of the `ttMigrate` operation that no object with the specified user was renamed. ■ If creating a database object with the new user returns an error, `ttMigrate` prints an error and continues restoring the rest of the database objects. ■ You cannot rename users `TTREP`, `SYS`, `SYSTEM` or `GRID`. ■ If you change a user which owns a table referenced by a view, you may need to drop and re-create the view. ■ If you change a user which owns a replicated table, you must rename that user on all replication peers in order to replicate updates successfully.
`-repUpgrade`	Ensures that the saved/restored tables are compatible for replication with the original tables when restoring tables for replication upgrade. This option is ignored when the `-c or -a` options are given. The `ttMigrate` utility in restore mode sets `-repUpgrade` by default. When restoring data from a `ttMigrate` datafile, if the file was created by a version of `ttMigrate` that does not support the `-repUpgrade` option, or if you explicitly specified`-noRepUpgrade` when you created the file (with a `ttMigrate` version before 5.1), you may encounter an error that indicates that a migration object could not be created with "replication compatibility." This option is set by default. If you use a version of `ttMigrate` that does not support the `-repUpgrade` option, or you explicitly specify `-noRepUpgrade` when creating a migration file, you may encounter an error that indicates that a migration object could not be created with "replication compatibility." You must either obtain a version of `ttMigrate` that supports the `-repUpgrade` option for the TimesTen release from which you are migrating, or, if supported, specify `-noRepUpgrade` when migrating files into a version of TimesTen that sets`-repUpgrade` by default.
`-saveAsCharset` *charset*	Allows you to save an object in a character set other than the database character set. When saving an object, `ttMigrate` stores it in the database character set by default.

Option	Description
`-updateStats`	Specifies that `ttMigrate` should update statistics on restored tables and materialized views. This option is ignored when the `-c or -a` options are given. If you specify both `-estimateStats` and `-updateStats`, statistics on restored tables are updated, not estimated. Use of this flag may improve the performance of materialized view restoration and may also improve the performance of queries on the restored tables and views.
`-v` *`verbosity`*	Specifies the verbosity level for messages printed when `ttMigrate` saves or restores a database. One of: 0 - Shows errors and warnings only. 1 - prints the name of each table as it is saved or restored. 2 - prints the name of each table or index as it is saved or restored. 3 - (the default) prints the name of each table or index as it is saved or restored and prints a dot (.) for each 10,000 rows saved or restored. -v is ignored in List, Long-list, Describe and Long-describe modes.
`-V | -version`	Prints the release number of `ttMigrate` and exits.
The following options are available in Restore mode (`-r`) only:	-
`-autorefreshPaused`	Restores cache groups with `AUTOREFRESH` attribute with autorefresh state paused. Otherwise the state is set to `OFF`.
`-C` *`chkPtFreq`*	Specifies that `ttMigrate` should checkpoint the database after restoring every *`chkPtFreq`* megabytes of data. A value of zero (the default) specifies that `ttMigrate` should never checkpoint the database.
`-convertCGTypes`	Determines the best type mapping from the underlying Oracle tables to TimesTen cached tables using: ■ The types of the columns in the Oracle tables ■ The types of the columns stored in the migration file and ■ The TimesTen-to-Oracle type mapping rules described above. If this option is specified with either the`-convertTypesToOra` or the `-convertTypesToTT` option, this option takes precedence for cached tables. This option does not impact non-cached tables.
`-localhost` *`hostName`*	Can be used to explicitly identify the name or IP address of the local host when restoring replicated tables.
`-noCharsetConversion`	Restores data in the database character set, not the tagged character set. See also `-saveAsCharset`. This option may be useful for legacy TimesTen users who may have migrated pre-11.2.1 data into a 11.2.1 or later release of TimesTen as TIMESTEN8 or another character set such as WE8ISO8895P1, when the data is actually in another character set. If, at a later time you wish to have that data interpreted according to its actual character set, use this option to migrate the data into a database that uses the data's actual character set with no character set conversion.
`-restorePublicPrivs`	Restores privileges that were granted to PUBLIC after the database was created. PUBLIC's privileges are not restored by default.

Modes

Create mode (-c) and Append mode (-a)

In Create mode, `ttMigrate` saves migrate objects from a TimesTen database into a new binary datafile. If the datafile does not exist, `ttMigrate` creates it. Otherwise, `ttMigrate` overwrites the existing file, destroying its contents.

The datafile format used by `ttMigrate` is independent of any release of TimesTen, so it is possible to use `ttMigrate` to migrate data from one TimesTen release to another.

In Append mode, `ttMigrate` appends migrate objects from a TimesTen database to an existing datafile. If the datafile does not exist, `ttMigrate` creates it.

For each ordinary (non-cached) table, `ttMigrate` saves:

- The table description: the name and type of each of the table's columns, including primary key and nullability information.
- The table's index definitions: the name of each index and the columns contained in the index. The actual contents of the index are not saved; `ttMigrate` only saves the information needed to rebuild the index when the table is restored.
- The table's foreign key definitions. You can disable the saving of foreign key definitions using the `-nf` option.
- The rows of the table. You can disable the saving of rows using the `-nr` option.

For each cache group, `ttMigrate` saves the following:

- The cache group definition: the cache group owner and name, the names of all tables in the cache group and any relevant cache group settings, such as the cache group duration.

  Note: After `ttMigrate` has been used to restore a database, all autorefresh cache groups in the restored database have `AUTOREFRESH STATE` set to `OFF`, no matter how it was set on the source database. Reset `AUTOREFRESH STATE` to `ON` by using the `ALTER CACHE GROUP` statement.

- All the cached tables in the cache group: the table name, column information, table attributes (propagate or read-only), `WHERE` clause, if any, foreign key definitions and index definitions.

For each view, `ttMigrate` saves the following:

- All the same information as a normal table.
- The query defining the view.

For each sequence, `ttMigrate` saves the following:

- The complete definition of the sequence.
- The sequence's current value.

For each user (except the instance administrator), `ttMigrate` saves the following:

- User name
- The user's encrypted password
- Privileges that have been granted to the user

For PUBLIC, ttMigrate saves all privileges that have been granted to PUBLIC after database creation.

If there are any replication schemes defined, ttMigrate saves all the of the TTREP tables containing the replication schemes. Replication schemes should have names that are unique from all other database objects. It is not possible to migrate a replication scheme with the same name as any other database object.

> **Note:** The ttMigrate utility does not save the rows of a cached table into the datafile, even if you have not specified the -nr option. The foreign key definitions of cached tables are always saved, regardless of the use of the -nf option, as they are needed to maintain the integrity of the cache group.

By default, ttMigrate saves all database objects and users in the database to the datafile, including tables, views, cache groups, sequences, users and replication schemes. Alternatively, you can give a list of database objects to be saved on the command line, except for replication schemes. The names in this list can contain the wildcard characters % (which matches one or more characters) and _ (which matches a single character). ttMigrate saves all database objects that match any of the given patterns. You do not need to be fully qualify names: If a name is given with no owner, ttMigrate saves all database objects that match the specified name or pattern, regardless of their owners.

You cannot save cached tables independently of their cache groups. If you list a cached table on the command line without also listing the corresponding cache group ttMigrate issues an error.

Use the -v option to control the information that ttMigrate prints while the save is in progress.

Restore mode (-r)

In Restore mode, ttMigrate restores all database objects from a datafile into a TimesTen database.

For each ordinary (non-cached) table, ttMigrate restores:

- The table, using the original owner, table name, column names, types and nullability and the original primary key.
- The table's foreign keys. You can use the -nf flag to disable the restoration of foreign keys.
- All indexes on the table.
- All rows of the table. You can use the -nr flag to disable the restoration of rows.

For each cache group, ttMigrate restores:

- The cache group definition, using the original cache group owner and name.
- Each cached table in the cache group, using the original table names, column names, types and nullability, the original primary key, the table attributes (PROPAGATE or READONLY), and the WHERE clause, if any.
- The foreign key definitions of the cached tables.
- All the indexes on the cached tables.

> **Note:** The `ttMigrate` utility does not restore the rows of cached tables, even if you have not specified the `-nr` option. The foreign key definitions of the cached tables are always restored, regardless of the use of the `-nf` option, as they are needed to maintain the integrity of the cache group.

By default, the `-repUpgrade` option is set during restore.

By default, `ttMigrate` restores all tables and cache groups in the datafile. Alternatively, you can list specific tables and cache groups to be restored on the command line. The names in this list must be fully qualified and cannot use wildcard characters.

You cannot restore cached tables independently of their cache groups. If you list a cached table on the command line without also listing the corresponding cache group, then `ttMigrate` issues an error.

Use the `-v` option to control the information that `ttMigrate` prints while the restoration is in progress.

The `-inline` option may be used to control whether variable length columns are restored as `INLINE` or `NOT INLINE`. See "Type specifications" in *Oracle TimesTen In-Memory Database SQL Reference*. In the default mode, `-inlinepreserve`, `ttMigrate` restores all variable-length columns with the same `INLINE` or `NOT INLINE` setting with which they were saved. In the other two modes, `-inlinedsDefault` and `-inline`*`maxlen`*, `ttMigrate` restores variable-length columns equal to or shorter than a threshold length as `INLINE`, and restores all other variable length columns as `NOT INLINE`. For`-inlinedsDefault`, this threshold is the default automatic `INLINE` length for a TimesTen database. The `-inline`*`maxlen`* mode restores variable length columns with a user-specified threshold length of *`maxlen`* as `INLINE`, and all other variable length columns as `NOT INLINE`, even if they were saved as `INLINE`. If *`maxlen`* is 0, then all variable-length columns are restored as `NOT INLINE`.

List mode (-l) and Long-list mode (-L)

In List mode, `ttMigrate` lists the names of database objects in the specified datafile, including cached tables and the replication scheme `TTREP` tables.

In Long-list mode, `ttMigrate` lists the names of database objects in the datafile, including cached tables and the replication scheme `TTREP` tables, along with the number of rows in each table and the index definitions for each table, the query defining each view and the specifications for each sequence.

By default, `ttMigrate` lists the replication scheme name and all the database objects in the file. Alternatively you can provide a list of names of database objects on the command line. The names in this list must be fully qualified and cannot use wildcard characters.

Describe mode (-d)

In Describe mode, `ttMigrate` gives a short description for database objects in the specified file.

For each table, `ttMigrate` lists the table name, the number of rows in the table, and the table's column definitions, primary key and foreign keys. For cached tables, `ttMigrate` also lists the table attributes (`PROPAGATE` or `READONLY`) and the table's `WHERE` clause, if any.

For views, `ttMigrate` also lists the query defining the view.

For cache groups, `ttMigrate` lists the cache group name, the number of tables in the cache group, the cache group duration and describes each cached table in the cache group.

For replication schemes, `ttMigrate` lists the replication scheme name and all the `TTREP` replication scheme tables in the same manner as user tables.

By default, `ttMigrate` describes all the database objects in the file. Alternatively, you can provide a list of names of database objects on the command line. The names in this list must be fully qualified and cannot use wildcard characters.

Long-describe mode (-D)

In Long-describe mode, `ttMigrate` gives a full description for database objects in the specified file.

For each table, `ttMigrate` lists the table's name and the number of rows in the table, the table's column definitions, primary key, foreign keys and index definitions. For cached tables, `ttMigrate` also lists the table attributes (`PROPAGATE` or `READONLY`) and the table's `WHERE` clause, if any.

For cache groups, `ttMigrate` lists the cache group name, the number of tables in the cache group, the cache group duration and describes each cached table in the cache group.

For sequences, `ttMigrate` lists all the values used to define the sequence and its current value.

For replication schemes, `ttMigrate` lists all the `TTREP` replication scheme tables in the same manner as user tables.

By default, `ttMigrate` describes all of database objects in the file. Alternatively, you can provide a list of names of database objects on the command line. The names in this list must be fully qualified and cannot use wildcard characters.

TimesTen to Oracle data type conversions

Both TimesTen and Oracle data types are supported in TimesTen 11.2.1 When migrating a database from an earlier version of TimesTen to TimesTen release 11.2.1, you can convert the data types in your database to the default Oracle type mode. This is not required, however.

In replication, the type mode must be the same on both sides of the replication scheme. Therefore you cannot convert the data types as part of an online upgrade, as TimesTen releases previous to 11.2.1 do not support Oracle data types.

Note: If `-convertTypesToOra` is specified, and a DECIMAL (or NUMERIC) column exists in the database with a precision > 38, the column is converted to a NUMBER column with a precision of 38, and a warning is returned. If this occurs, and column values exist that overflow or underflow with a precision of 38, those values are reduced or increased to the maximum or minimum possible value for a NUMBER with a precision of 38. Because of this and some other cases, the data type conversion procedures (using `-convertTypesToOra` and `-convertTypesToTT`) are not guaranteed to be reversible. Converting types from TT->ORA->TT can result in columns and data which are different from the original in some cases.

To convert from TimesTen data types to Oracle data types, use the `-convertTypesToOra` option.

The `-convertTypesToOra` option instructs `ttMigrate` to make the following type conversions as it saves or restores tables:

From TimesTen Type	To Oracle Type
TT_CHAR	ORA_CHAR
TT_VARCHAR	ORA_VARCHAR2
TT_NCHAR	ORA_NCHAR
TT_NVARCHAR	ORA_NVARCHAR2
TT_DECIMAL	ORA_NUMBER
TT_DATE	ORA_DATE (append 12:00:00 am)
TT_TIMESTAMP	ORA_TIMESTAMP(6)

Note: Columns of type TT_TINYINT, TT_SMALLINT, TT_INTEGER, TT_BIGINT, BINARY_FLOAT, BINARY_DECIMAL, TT_BINARY, TT_VARBINARY, and TT_TIME are not converted.

For information on data types, see "Data Types" in the *Oracle TimesTen In-Memory Database SQL Reference*.

Oracle to TimesTen data type conversions

When migrating tables backward from TimesTen release 11.2.1 to an earlier version of TimesTen, you may need to convert Oracle data types to TimesTen data types, as the Oracle data types were not supported in releases before 11.2.1.

To convert from Oracle data types to TimesTen data types, use the `-convertTypesToTT` option.

The `-convertTypesToTT` option instructs the `ttMigrate` utility to make the following type conversions as it saves or restores tables:

From Oracle Type	To TimesTen Type
ORA_CHAR	TT_CHAR
ORA_VARCHAR2	TT_VARCHAR
ORA_NCHAR	TT_NCHAR
ORA_NVARCHAR2	TT_NVARCHAR
ORA_NUMBER	TT_DECIMAL
ORA_DATE	TT_DATE (time portion of date is silently truncated)
ORA_TIMESTAMP	TT_TIMESTAMP

For information on data types, see "Data Types" in the *Oracle TimesTen In-Memory Database SQL Reference*.

Cache group data type conversions

When restoring a database that contains cache groups from a TimesTen release that is earlier than 7.0, use the `-convertCGTypes`. option to convert the data type of columns from pre-7.0 types to more clearly map with the data types of the columns in the Oracle database with which the cache group is associated.

The following table describes the type mapping.

Pre-7.0 TimesTen Type	Oracle Type	Converted Type
TINYINT	NUMBER(p,s) when s > 0	NUMBER(p,s)
TINYINT	NUMBER(p,s) when s <= 0	TT_TINYINT
SMALLINT	NUMBER(p,s) when s > 0	NUMBER(p,s) TT_SMALLINT
SMALLINT	NUMBER(p,s) when s <= 0	TT_SMALLINT
INTEGER	NUMBER(p,s) when s > 0	NUMBER(p,s)
INTEGER	NUMBER(p,s) when s <= 0	TT_INTEGER
BIGINT	NUMBER(p,s) when s > 0	NUMBER(p,s)
BIGINT	NUMBER(p,s) when s <= 0	TT_BIGINT
NUMERIC(p,s)DECIMAL(p,s)	NUMBER	NUMBER
NUMERIC(p,s)DECIMAL(p,s)	NUMBER(x,y)	NUMBER(x,y)
NUMERIC(p,s)DECIMAL(p,s)	FLOAT(x)	NUMBER(p,s)
REAL	Any	BINARY_FLOAT
DOUBLE	Any	BINARY_DOUBLE
FLOAT(x) x <=24	Any	BINARY_FLOAT
FLOAT(x) x >= 24	Any	BINARY_DOUBLE
CHAR(x)	Any	ORA_CHAR(x)
VARCHAR(x)	Any	ORAVARCHAR2(x)
BINARY(x)	Any	TT_BINARY(x)
VARBINARY(x)	Any	TT_VARBINARY(x)
DATE	DATE	ORA_DATE
TIMESTAMP	DATE	ORA_DATE
TIME	DATE	ORA_DATE
Any1	TIMESTAMP(m)	ORA_TIMESTAMP(m)

Note: "Any" means the type value does not affect the converted result type.

For information on data types, see "Data Types" in the *Oracle TimesTen In-Memory Database SQL Reference* and "Mappings between Oracle and TimesTen data types" in the *Oracle In-Memory Database Cache User's Guide*.

Return codes

The `ttMigrate` utility restore (`-r`) and create (`-c`) commands return these exit codes:

0 - All objects were successfully created or restored.

1 - Some objects successfully created or restored. Some objects could not be created or restored due to errors.

2 - Fatal error, for example, could not connect or could not open the data file.

3 - CTRL-C or another signal received during the create or restore operation.

Examples

The following command dumps all database objects from database `SalesDS` into a file called `sales.ttm`. If `sales.ttm` exists, `ttMigrate` overwrites it.

```
ttMigrate -c DSN=SalesDS sales.ttm
```

This command appends all database objects in the `SalesDS` database owned by user `MARY` to `sales.ttm`:

```
ttMigrate -a DSN=SalesDS sales.ttm MARY.%
```

This command restores all database objects from `sales.ttm` into the `SalesDS` database:

```
ttMigrate -r DSN=SalesDS sales.ttm
```

This command restores `MARY.PENDING` and `MARY.COMPLETED` from `sales.ttm` into `SalesDS` (Case is ignored in migrate objects.):

```
ttMigrate -r DSN=SalesDS sales.ttm MARY.PENDINGMARY.COMPLETED
```

This command lists all migrate objects saved in `sales.ttm`:

```
ttMigrate -l sales.ttm
```

Notes

When migrating backward into a release of the Oracle TimesTen In-Memory Database that does not support features in the current release, TimesTen generally issues a warning and continues without migrating the unsupported features. In a few cases, where objects have undergone conversion, `ttMigrate` may fail and return an error message. This may be the case with conversions of data types, character sets and primary key representation.

The following restrictions, limitations and suggestions should be considered before preparing to use `ttMigrate`.

Asynchronous materialized view: When migrating to a previous release, asynchronous materialized views are ignored and TimesTen returns a warning.

Cache groups: In restore mode, the presence of foreign key dependencies between tables may require `ttMigrate` to reorder tables to ensure that a child table is not restored before a parent table.

When migrating databases that contain cache groups from a previous release of TimesTen to TimesTen 7.0 or greater, you must use the option `-convertTypesToOra`. See "Cache group data type conversions" on page 3-82 for a description of the data type mapping.

Character columns in cached tables must have not only the same length but also the same byte semantics as the underlying Oracle tables. Cache group migration fails when there is a mismatch in the length or length semantics of any of its cached tables.

Character sets: By default, `ttMigrate` stores table data in the database character set, unless you have specified the `-saveAsCharset` option. At restore time, conversion to another character set can be achieved by migrating the table into a database that has a different database character set. When migrating data from a release of TimesTen that is earlier than 7.0, TimesTen assumes that the data is in the target database's character set. If the data is not in the same database character set as the target database, the data may not be restored correctly.

When migrating columns with BYTE length semantics between two databases that both support NLS but with different database character sets, it is possible for migration to fail if the columns in the new database are not large enough to hold the values in the migrate file. This could happen, for example, if the source database uses a character set whose maximum byte-length is 4 and the destination database uses a character set whose maximum byte-length is 2.

TimesTen issues a warning whenever character set conversion takes place to alert you to the possibility of data loss due to conversion.

Data type conversions: When migrating data from a pre-7.0 release of TimesTen, you must explicitly request data type conversions, using either the `-convertTypesToOra` or the `-convertTypesToTT` options.

ttMigrate saves the length semantic annotation (BYTE or CHAR) of CHAR and VARCHAR columns and restores these annotations when restoring into TimesTen releases that support them. When migrating backward into a TimesTen release that does not support these annotations, columns with CHAR length semantics are converted to BYTE length, but their lengths are adjusted to match the byte length of the original columns. When migrating forward from a release that does not support these annotations, BYTE length semantics are used.

Foreign key dependencies: In restore mode, the presence of foreign key dependencies between tables may require `ttMigrate` to reorder tables to ensure that a child table is not restored before any of its parents. Such dependencies can also prevent a child table from being restored if any of its parent tables were not restored. For example, when restoring a table A that has a foreign key dependency on a table B, `ttMigrate` first checks to verify that table B exists in the database. If table B is not found, `ttMigrate` delays the restoration of table A until table B is restored. If table B is not restored as part of the `ttMigrate` session, TimesTen prints an error message indicating that table A could not be restored due to an unresolved dependency.

Indexes: TimesTen supports range indexes as primary-key indexes into TimesTen releases that support this feature. When migrating backward into a release that does not support range indexes as primary-key indexes, the primary keys are restored as hash indexes of the default size. When migrating forward from a release that does not support range indexes as primary-key indexes, the primary keys are restored as hash indexes of the same size as the original index.

TimesTen also supports bitmap indexes. When migrating backward into a release that does not support bitmap indexes, `ttMigrate` converts the bitmap indexes to range indexes.

INLINE columns: When migrating TimesTen tables that contain `INLINE` variable length columns to a release of TimesTen that is earlier than 5.1, you must explicitly use the `-noRepUpgrade` option. Using the default `-repUpgrade` option results in an error. The `INLINE` attribute of the columns is maintained, unless you specify otherwise using the `-inline` option.

Materialized view logs: TimesTen does not save the content of materialized view logs, only the definition.

Replication: Before attempting a full store migrate of replicated stores, make sure the host name and database name are the same for both the source and destination databases.

System views: TimesTen does not save the definitions or content of system vies during migration.

Other considerations: Because `ttMigrate` uses a binary format, you cannot use `ttMigrate` to:

- Migrate databases between hardware platforms.
- Restore data saved with `ttBackup` or use `ttBackup` to restore data saved with `ttMigrate`.

Platforms: You can use `ttMigrate` to migrate databases between 32- and 64-bit platforms if the two platforms are otherwise the same. Follow the examples in the *Oracle TimesTen In-Memory Database Installation Guide*.

WINDOWS

UNIX

- On Windows, you can use `ttMigrate` to access databases from any release of TimesTen. On Windows, this utility is supported for all TimesTen Data Manager and Client DSNs.
- On UNIX, the release of `ttMigrate` must match the release of the database you are connecting to.

See also

`ttBackup`
`ttBulkCp`
`ttRestore`

ttmodinstall

Description

Modifies specified settings for an installation.

Required privilege

This utility requires the instance administrator privilege.

Syntax

```
ttmodinstall {-h | -help | -?}
ttmodinstall {-V | -version}
ttmodinstall -port portNumber
ttmodinstall -tns_admin path
ttmodinstall -enablePLSQL
ttmodinstall -crs
```

Options

`ttmodinstall` has the options:

Option	Description

`-h`
`-help`
`-?`
	Displays help information.

`-crs`	Create or modify Oracle Clusterware configuration. For more information, see "Using Oracle Clusterware to Manage Active Standby Pairs" in *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*.
`-enablePLSQL`	Enables PL/SQL in the database.
`-port` *`portNumber`*	Changes the daemon port for the current instance of TimesTen to *`portNumber`*. This is useful if you discover that other processes are listening on the port that you assigned to TimesTen at installation time. Also can be used to assign the port for the TimesTen cluster agent. See "Using Oracle Clusterware to Manage Active Standby Pairs" in *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*.
`-tns_admin` *`path`*	Sets the value for the TNS_ADMIN environment variable. Specify the directory where the `tnsnames.ora` file can be found.
`-V | -version`	Display TimesTen version information.

Examples

To change the port number of the TimesTen instance to 12345, use:

```
ttmodinstall -port 12345
```

Notes

All TimesTen operations must be completely shut down to use this utility. This utility stops and then restarts the TimesTen daemon before making any changes to the instance.

ttRepAdmin

Description

Displays existing replication definitions and monitors replication status. The `ttRepAdmin` utility is also used when upgrading to a new release of TimesTen, as described in *Oracle TimesTen In-Memory Database Installation Guide*.

Required privilege

This utility requires the `ADMIN` privilege.

Syntax

```
ttRepAdmin {-h | -help | -?}
ttRepadmin {-V | -version}
ttRepAdmin -self -list [-scheme [owner.]schemeName]
       {DSN | -connStr connectionString}

ttRepAdmin -receiver [-name receiverName]
      [-host receiverHostName] [-state receiverState] [-reset]
      [-list] [-scheme [owner.]schemeName]
      {DSN | -connStr connectionString}

ttRepAdmin -log {DSN | -connStr connectionString}

ttRepAdmin -showstatus {-awtmoninfo} {DSN | -connStr connectionString}

ttRepAdmin -showconfig {DSN | -connStr connectionString}

ttRepAdmin -bookmark {DSN | -connStr connectionString}

ttRepAdmin -wait [-name receiverName] [-host receiverHostName]
      [-timeout seconds] {DSN | -connStr connectionString}

ttRepAdmin -duplicate -from srcDataStoreName
      -host srcDataStoreHost
      [-setMasterRepStart] [-ramLoad] [-delXla]
      [-UID userId] [-PWD pwd | -PWDCrypt encryptedPwd]
      [-drop { [owner.]table ... | [owner.]sequence |ALL }]
      [-truncate { [owner.]table ... | ALL }]
      [-compression 0 | 1] [-bandwidthmax maxKbytesPerSec]
      [-initCacheDr [-noDRTruncate]]
      [-keepCG [-cacheUid cacheUid [-cachePwd cachePwd]]
         [-recoveringNode | -deferCacheUpdate]
      | -nokeepCG]
      [-remoteDaemonPort portNo] [-verbosity {0|1|2}]
      [-localhost localHostName]
      {destDSN | -connStr connectionString}
```

ttRepAdmin operations

The `ttRepAdmin` utility is used for many replication operations. These operations fall into the following categories:

- Help and version information
- Database information
- Subscriber database operations

- Duplicate a database
- Wait for updates to complete
- Replication status

Help and version information

Use this form of `ttRepAdmin` to obtain help and the current version of TimesTen.

```
ttRepAdmin {-h | -help | -?}
ttRepadmin {-V | -version}
```

Option	Description

`-h`
`-help`
`-?`
	Display help information.

`-V | -version`	Display TimesTen version information.

Database information

Use this form of `ttRepAdmin` to obtain summary information about a database.

```
ttRepAdmin -self -list [-scheme [owner.]schemeName]
{DSN | -connStr connectionString}
```

Options

`ttRepAdmin -self -list` has the options:

Option	Description
DSN	Data source name of a master or subscriber database.
`-connStr` *connection_string*	Connection string of a master or subscriber database.
`-self`	Specified database.
`-list`	Lists database name, host, port number, and bookmark position.
`-scheme` [*owner*.]*schemeName*]	Name of replication scheme when there is more than one scheme.

Examples

```
ttRepAdmin -self -list my_dsn
```

The above syntax prints out information about the replication definition of the database `my_dsn`.

Subscriber database operations

Use this form of `ttRepAdmin` to check the status or reset the state of a subscriber (receiver) database.

```
ttRepAdmin -receiver [-name receiverName]
[-host receiverHostName]
       [-state receiverState] [-reset]
       [-list] [-scheme [owner.]schemeName]
       {DSN | -connStr connectionString}
```

Options

`ttRepAdmin -receiver` has the options:

Option	Description
DSN	Data source name of the master database.
`-connStr` *connection_string*	Connection string of the master database.
`-receiver`	Subscriber databases receiving updates from the master. Use `-name` and `-host` to specify a specific subscriber database.
`-name` *receiverName*	A specific subscriber (receiving) database. The receiverName is the last component in the database path name.
`-host` *receiverHostName*	Hostname or TCP/IP address of the subscriber host.

`-state start`
`-state stop`
`-state pause`
	Sets the state of replication for the subscriber.
	▪ `start` - Starts replication to the subscriber (default).
	▪ `stop` - Stops replication to the subscriber, discarding updates.
	▪ `pause` - Pauses the replication agent, preserving updates.
	See "Setting the replication state of subscribers" in the *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide* for more information.

`-reset`	Clears the bookmark in the master database log for the latest transaction to be sent to a given subscriber. This option should only be used when the transaction numbering of the master database is changed, such as when the database is re-created using `ttMigrate` or `ttBackup`. If the master database is saved and restored using `ttBackup` and `ttRestore`, transaction numbering is preserved and this option should not be used.
`-list`	Lists information about a replication definition.
`-scheme` [*owner.*]*schemeName*]	Specifies the replication scheme name when there is more than one scheme.

Examples

```
ttRepAdmin -receiver -list my_dsn
```

The above syntax lists replication information for all the subscribers of the master database, `my_dsn`.

```
ttRepAdmin -receiver -name rep_dsn -list my_dsn
```

The above syntax lists replication information for the `rep_dsn` subscriber of the master database, `my_dsn`.

```
ttRepAdmin -receiver -name rep_dsn -reset my_dsn
```

The above syntax resets the replication bookmark with respect to the `rep_dsn` subscriber of the master database. Should only be used when migrating a replicated database with `ttMigrate` or `ttBulkCp`.

```
ttRepAdmin -receiver -name rep_dsn -state Start my_dsn
```

The above syntax resets the replication state of the `rep_dsn` subscriber database to the Start state with respect to the master database, `my_dsn`.

Duplicate a database

Use this form of `ttRepAdmin` to create a new database with the same contents as the master database.

The following must be true for you to perform the `ttRepAdmin -duplicate`:

- Only the instance administrator can run `ttRepAdmin -duplicate`.
- The instance administrator must have the same operating system username on both source and target computer to execute `ttRepAdmin -duplicate`.
- You must provide the user name and password with the `-UID` and `-PWD` options for a user with the ADMIN privilege on the source database.

Before running the `ttRepAdmin -duplicate` command, use `ttStatus` to be sure that the replication agent is started for the source database.

```
ttRepAdmin -duplicate -from srcDataStoreName
       -host srcDataStoreHost
      [-setMasterRepStart] [-ramLoad] [-delXla]
      -UID userId (-PWD pwd | -PWDCrypt encryptedPwd)
      [-drop { [owner.]table ... | [owner.]sequence |ALL }]
      [-truncate { [owner.]table ... | ALL }]
      [-compression 0 | 1] [-bandwidthmax maxKbytesPerSec]
      [-initCacheDr [-noDRTruncate]]
      [-keepCG [-cacheUid cacheUid [-cachePwd cachePwd]]
      [-recoveringNode | -deferCacheUpdate]
      |-nokeepCG]
      [-remoteDaemonPort portNo] [-verbosity {0|1|2}]
      [-localhost localHostName]
      {destDSN | -connStr connectionString}
```

Options

`ttRepAdmin -duplicate` has the options:

Option	Description
`-bandwidthmax` *`maxKbytesPerSec`*	Specifies that the duplicate operation should not put more than *`maxKbytesPerSec`* KB of data per second onto the network. A value of 0 indicates that there should be no bandwidth limitation. The default is 0. The maximum is 9999999.
`-compression 0 | 1`	Enables or disables compression during the duplicate operation. The default is 0 (disabled).
`-connStr` *`connection_string`*	Specifies the connection string of the destination database.
`-delXla`	Removes all the XLA bookmarks as part of the duplicate operation. Use this option if you do not want to copy the bookmarks to the duplicate database.
`destDSN`	Indicates the data source name of the destination database.
`-drop {[owner.]`*`table`*`... |[`*`owner`*`.]`*`sequence`*`|ALL`	Drops any tables or sequences that are copied as part of the -duplicate operation but which are not included in the replication scheme. The option is ignored if the table is a cache group table.

Option	Description
`-duplicate`	Creates a duplicate of the specified database using replication to transmit the database contents across the network. See "Duplicating a database" in *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*.
`-from` *`srcDataStoreName`*	Used with `-duplicate` to specify the name of the sender (or master) database. The *`srcDataStoreName`* is the last component in the database path name.
`-host` *`srcDataStoreName`*	Defines the hostname or TCP/IP address of the sender (or master) database.
`-initCacheDr`	Initializes disaster recovery. Must be used with `-cacheUid` and `-cachePwd` options.
`-keepCG [-cacheUid` `cacheUid -cachePwd` `cachePwd]` `[-recoveringNode |` `-deferCacheUpdate] |` `-noKeepCG`	`-keepCG` and `-noKeepCG` specify whether tables in cache groups should be maintained as cache group tables or converted to regular tables in the target database. The default is `-noKeepCG`. `cacheUid` is the cache administration user ID. `cachePwd` is the password for the cache administrator user. If no password is provided, `ttRepAdmin` prompts for a password. `-recoveringNode` specifies that `-duplicate` is being used to recover a failed node for a replication scheme that includes an AWT or autorefresh cache group. Do not specify `-recoveringNode` when rolling out a new or modified replication scheme to a node. If `-duplicate` cannot update metadatabase on the Oracle database and all incremental autorefresh cache groups are replicated, then updates to the metadata are automatically deferred until the cache and replication agents are started. `-deferCacheUpdate` forces the deferral of changes to metadatabase on the Oracle database until the cache and replication agents are started and the agents can connect to the Oracle database. Using this option can cause a full autorefresh if some of the incremental cache groups are not replicated or if `-duplicate` is being used for rolling out a new or modified replication scheme to a node.
`-localhost` *`hostName`*	Can be used with `-duplicate` and`-setMasterRepStart` to explicitly identify the name or IP address of the local host.
`-noDRTruncate`	Used with the `-initCacheDr` option, `-noDRTruncate` disables truncation of Oracle tables during the initial rollout process for the remote subscriber on the Disaster Recovery site. When `-noDRTruncate` is specified, TimesTen does not truncate the Oracle database tables that correspond to the asynchronous writethrough cache group tables in an active standby pair replication scheme.
`-PWD` *`pwd`*	The password of the user specified in the `-UID` option.
`-PWDCrypt` *`encryptedPwd`*	The encrypted password of the user specified in the `-UID` option.
`-ramLoad`	Keeps the database in memory upon completion of the duplicate operation. This option avoids the unload/reload database cycle to improve the performance of the duplicate operation when copying large databases. After the duplicate option, the RAM Policy for the database is set to "manual." Use the `ttAdmin` utility to make further changes to the RAM policy.

Option	Description
`-remoteDaemonPort` *`portNo`*	The port number of the remote main daemon. The port number supplied as an argument to this option is used unless the value is zero. In that case the default behavior to determine the port number is used. The `-remoteDaemonPort` option cannot be used to duplicate databases that have stores which use automatic port configuration.
`-setMasterRepStart`	When used with `-duplicate`, this option sets the replication state for the newly created database to the Start state just before the database is copied across the network. This ensures that all updates made to the source database after the duplicate operation are replicated to the newly duplicated local database. Any unnecessary transaction log files for the database are removed.
`-truncate [`*`owner.`*`]`*`table`* `...| ALL`	Truncates any tables that are copied as part of the `-duplicate` operation but which are not included in the replication scheme. The option is ignored if the table is a cache group table.
`-UID` *`userid`*	The user ID of a user having the ADMIN privilege on the source database must be supplied.
`-verbosity {0 | 1 | 2}`	Provide details of the communication steps within the duplicate process and reports progress information about the duplicate transfer. `0` - No diagnostics are returned (default). `1` - Reports details of the duplicate parameters to `stdout`. `2` - Reports details of the duplicate parameters and details of the duplicate transfer operation to `stdout`.

Examples

Example 3–1 Duplicating a database

On the source database, create a user and grant the ADMIN privilege to the user:

```
CREATE USER ttuser IDENTIFIED BY ttuser;
User created.

GRANT admin TO ttuser;
```

The instance administrator must have the same user name on both instances involved in the duplication. Logged in as the instance administrator, duplicate the `ds1` database on `server1` to the `ds2` database:

```
ttRepAdmin -duplicate -from ds1 -host "server1"
           -UID ttuser -PWD ttuser
           -connStr "dsn=ds2;UID=ttuser;PWD=ttuser"
```

Example 3–2 Duplicating a database with cache groups

Use the `-keepCG` option to keep cache group tables when you duplicate a database. Specify the cache administration user ID and password with the `-cacheuid` and `-cachepwd` options. If you do not provide the cache administration user password, `ttRepAdmin` prompts for a password.

If the cache administration user ID is `orauser` and the password is `orapwd`, duplicate database `dsn1` on `host1`:

```
ttRepAdmin -duplicate -from dsn1 -host host1 -uid ttuser -pwd ttuser
        -keepCG -cacheuid orauser -cacheuid orapwd "DSN=dsn2;UID=;PWD="
```

The `UID` and `PWD` for `dsn2` are specified as null values in the connection string so that the connection is made as the current OS user, which is the instance administrator. Only the instance administrator can run `ttRepAdmin -duplicate`. If `dsn2` is configured with `PWDCrypt` instead of `PWD`, then the connection string should be `"DSN=dsn2;UID=;PWDCrypt="`.

Example 3–3 Setting the replication state on the source database

The `-setMasterRepStart` option causes the replication state in the *`srcDataStoreName`* database to be set to the `Start` state before it is copied across the network and then keeps the database in memory. It ensures that any updates made to the master after the duplicate operation has started are copied to the subscriber.

You can use the `-localhost` option to identify the local host by host name or IP address. These options ensure that all updates made after the duplicate operation are replicated from the remote database to the newly created or restored local database.

```
ttRepAdmin -duplicate -from srcDataStoreName -host srcDataStoreHost
        -setMasterRepStart -ramLoad
        -UID timesten_user -PWD timesten_user]
        -localhost localHostName
        [destDSN | -connStr connectionString ]
```

Notes

This utility can duplicate any temporary table definition in a database, but it does not replicate the contents of temporary tables.

You cannot use this utility to duplicate databases across major releases of TimesTen.

Wait for updates to complete

Use this form of `ttRepAdmin` to assure that all the updates in the log are replicated to all subscribers before call returns.

```
ttRepAdmin -wait [-name receiverName] [-host receiverHostName]
[-timeout seconds] {DSN | -connStr connectionString}
```

Options

`ttRepAdmin -wait` has the options:

Option	Description
DSN	Indicates the data source name of the master database.
`-connStr` *connection_string*	Specifies the connection string of the master database.
`-wait`	Waits for replication to become current before continuing.
`-name` *receiverName*	Identifies the database. The database name is the last component in the database path name.
`-host` *receiverHostName*	Defines the hostname or TCP/IP address of the subscriber host.
`-timeout` *seconds*	Specifies timeout value in seconds. `ttRepAdmin` returns within this amount of time, even if all updates to subscribers have not been completed.

Examples

```
ttRepAdmin -wait -name receiverName -host receiverHostName
-timeout seconds -dsn DSN
```

The above syntax provides a way to ensure that all updates, committed at the time this program was invoked, have been transmitted to the subscriber, *receiverName*, and the subscriber has acknowledged that all those updates have been durably committed at the subscriber database. The timeout in seconds limits the wait.

> **Note:** If `ttRepAdmin -wait` is invoked after all write transaction activity is quiesced at a store (there are no active transactions and no transactions have started), it may take 60 seconds or more before the subscriber sends the acknowledgment that all updates have been durably committed at the subscriber.

```
ttRepAdmin -wait -dsn DSN
```

In the above syntax, if no timeout and no subscriber name are specified, `ttRepAdmin` does not return until all updates committed at the time this program was invoked have been transmitted to all subscribers and all subscribers have acknowledged that all those updates have been durably committed at the subscriber database.

Replication status

Use this form of `ttRepAdmin` to check the size of the transaction log files, bookmark position, or replication configuration of a master database.

```
ttRepAdmin -log {DSN | -connStr connectionString}
ttRepAdmin -showstatus {-awtmoninfo} {DSN | -connStr connectionString}
ttRepAdmin -showconfig {DSN | -connStr connectionString}
ttRepAdmin -bookmark {DSN | -connStr connectionString}
```

Options

The `ttRepAdmin` monitor operations have the options:

Option	Description
DSN	Indicates the data source name of the master database.
`-awtmoninfo`	If you have enabled monitoring for AWT cache groups by calling the `ttCacheAwtMonitorConfig` procedure, you can display the monitoring results by using the this option. If AWT monitoring is enabled, `ttrepadmin -awtmoninfo` displays the output: ▪ TimesTen processing time: The total number of milliseconds spent in processing AWT transaction data since monitoring was enabled. ▪ Oracle bookmark management time: The total number of milliseconds spent in managing AWT metadata on Oracle since monitoring was enabled.
`-connStr` *connection_string*	Specifies the connection string of the master database.
`-log`	Prints out number and size of transaction log files retained by replication to transmit updates to other databases.
`-showconfig`	Lists the entire replication configuration. See "Show configuration of replicated databases" in the *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide* for more information.
`-showstatus`	Reports the current status of the specified replicated database. See "Show replication status" in the *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide* for more information.
`-bookmark`	Reports the latest marker record from where replication must read the log, the most recently created log sequence number, and the latest log sequence number whose record has been flushed to disk. Bookmarks are not supported if you have configured parallel replication. See "Show replicated log records" in the *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide* for more information.

Result set

If AWT monitoring is enabled, this utility displays the following information in addition to other `ttRepAdmin -showstatus` output:

- TimesTen processing time: The total number of milliseconds spent in processing AWT transaction data since monitoring was enabled.
- Oracle bookmark management time: The total number of milliseconds spent in managing AWT metadata on Oracle since monitoring was enabled.
- Oracle execute time: The total number of milliseconds spent in OCI preparation, binding and execution for AWT SQL operations since monitoring was enabled. This statistic includes network latency between TimesTen and Oracle.
- Oracle commit time: The total number of milliseconds spent in committing AWT updates on Oracle since monitoring was enabled. This statistic includes network latency between TimesTen and Oracle.
- Time since monitoring was started.
- Total number of TimesTen row operations: The total number of rows updated in AWT cache groups since monitoring was enabled.
- Total number of TimesTen transactions: The total number of transactions in AWT cache groups since monitoring was enabled.
- Total number of flushes to Oracle: The total number of times that TimesTen data has been sent to Oracle.

The output also includes the percentage of time spent on TimesTen processing, Oracle bookmark management, Oracle execution and Oracle commits.

Examples

```
ttRepAdmin -log DSN
```

The above syntax reports the number of transaction log files that replication is retaining to transmit updates to other databases. The replication agent retains a transaction log file until all updates in that transaction log file have been successfully transferred to each subscriber database.

```
ttRepAdmin -showconfig DSN
```

The above syntax reports the entire replication configuration. It lists all the subscribers for the specified DSN, the names and details of the tables being replicated, and all the subscriptions.

```
ttRepAdmin -showstatus DSN
```

The above syntax reports the current state of the database for the specified DSN. The output includes the state of all the threads in the replication agents for the replicated databases, bookmark locations, port numbers, and communication protocols.

```
ttRepAdmin -bookmark DSN
```

The above syntax prints out the log sequence numbers of the earliest log record still needed by replication, the last log record written to disk, and the last log record generated.

```
ttRepAdmin -showstatus -awtmoninfo myDSN

[other -showstatus output]
...
AWT Monitoring statistics
-------------------------
TimesTen processing time : 0.689000 millisecs (0.164307 %)
   Oracle bookmark management time : 3.229000 millisecs (0.770027%)
```

```
Oracle execute time : 342.908000 millisecs (81.774043 %)
Oracle commit time : 72.450000 millisecs (17.277315 %)
Time since monitoring was started: 8528.641000 millisecs
Cache-connect Operational Stats :
   Total Number of TimesTen row operations : 2
   Total Number of TimesTen transactions : 2
   Total Number of flushes to Oracle : 2
```

The above syntax and output shows the AWT monitoring status.

Notes

The `ttRepAdmin` utility is supported only for TimesTen Data Manager DSNs. It is not supported for TimesTen Client DSNs.

You must use the `-scheme` option when specifying more than one replication scheme, or when more than one scheme exists involving the specified database.

Using SQL configuration, you can create multiple replication schemes in the same database. If there is only one replication scheme, the `ttRepAdmin` utility automatically determines the scheme. If there is more than one scheme, you must use the `ttRepAdmin -scheme` option to specify which scheme to use.

When configuring replication for databases with the same name on different hosts, you can indicate which database you wish to operate on by using `-host`. For example, if all the subscribers have the name DATA, you can set the replication state on host `SW1` with:

```
ttRepAdmin -receiver -name DATA -host SW1 -state start DSN
```

See also

For a full description of TimesTen Replication, see *Oracle TimesTen In-Memory Database TimesTen to TimesTen Replication Guide*.

For upgrade examples, see "Database Upgrades" in *Oracle TimesTen In-Memory Database Installation Guide*.

ttRestore

Description

Creates a database from a backup that has been created using the `ttBackup` utility. If the database exists, `ttRestore` does not overwrite it.

The attributes in the `ttRestore` connection string can contain any of the First Connection or General Connection attributes. It can also include the Data Store attribute `LogDir`. All other Data Store attributes are copied from the backup files. This allows the restored database to be relocated.

The `ttRestore` action is somewhat more powerful than a first connect, as it can move the database. It is somewhat less powerful than creating a new database, as it cannot override the other Data Store attributes.

For an overview of the TimesTen backup and restore facility, see Migration, backup, and restoration of the database" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This utility requires the instance administrator privilege.

Syntax

```
ttRestore {-h | -help | -?}
ttRestore {-V | -version}
ttRestore [-fname filePrefix] [-noconn] -dir directory
       {DSN | -connStr connectionString}
ttRestore -i [-noconn] {DSN | -connStr connection_String}
```

Options

`ttRestore` has the options:

Option	Description
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the database to be administered.
`-dir` *`directory`*	Specifies the directory where the backup files are stored.
`-fname` *`filePrefix`*	Specifies the file prefix for the backup files in the backup directory. The backup files must have been stored in the backup directory with this prefix. The default value for this parameter is the file name portion of the `DataStore` parameter of the database's ODBC definition.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-i`	Read standard input for the backup data. You cannot use the `-dir` or `-fname` options with `-i`. `ttRestore` rolls the logs forward.

Option	Description
`-noconn`	In order to ensure that the restore was successful, `ttRestore` connects to the database as a last step. This option disables that last connect.
`-V | -version`	Prints the release number of `ttRestore` and exits.

Example

```
ttRestore -dir /users/pat/TimesTen/backups
-fname FastInsBkup "DSN=FastIns"
```

To back up a database named `origDSN` to the directory `/users/rob/tmp` and restore it to database named `restoredDSN`, use:

```
ttBackup -dir /users/rob/tmp -fname restored "dsn=origDSN"
ttRestore -dir /users/rob/tmp -fname restored "dsn=restoredDSN"
```

The value of `fname` is the name that you want for the prefix portion of the backup file name.

UNIX

On UNIX, to restore a tape backup to the `FastIns` database, use:

```
dd bs=64k if=/dev/rmt0 | ttRestore -i DSN=FastIns
```

Note

This utility is supported only where the TimesTen Data Manager is installed.

See also

`ttBackup`
`ttBulkCp`
`ttMigrate`

ttSchema

Description

Prints out the schema, or selected objects, of a database. The utility can list the following schema objects that are found in SQL `CREATE` statements:

- Tables
- Indexes
- Cache group definitions
- Sequences
- Views
- Materialized view logs
- Column definitions, including partition information
- PL/SQL program units

The level of detail in the listing and the objects listed are controlled by options. The output represents a point in time snapshot of the state of a database rather than a history of how the database came to arrive at its current state, perhaps through `ALTER` statements. An entire database, including data, cannot be completely reconstructed from the output of `ttSchema`. The output of `ttSchema` can be played back by the `ttIsql` utility in order to rebuild the full schema of a database.

Required privilege

This utility requires no privileges beyond those needed to perform describe operations on database objects.

This utility prints information only about the objects owned by the user executing the utility, and those objects for which the owner has `SELECT` privileges. If the owner executing the utility has `ADMIN` privilege, `ttSchema` prints information about all objects.

Syntax

```
ttSchema {-h | -help | -?}
ttSchema {-V | -version}
ttSchema [-l] [-c] [-fixedTypes] [-st | -systemTables]
         [ -list {all | tables | views | sequences |
         cachegroups | repschemes | plsql} [,...] ]
         [-plsqlAttrs | -noplsqlAttrs]
         [-plsqlCreate |-plssqlCreateOrReplace]
         {-connStr connection_string | DSN }
         [[owner.]object_name][...]
```

Options

`ttSchema` has the options:

Option	Description
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.

Option	Description
`-c`	Compatibility mode. Limits the use of TimesTen-specific and release-specific keywords and extensions. This may be useful if the `ttSchema` output is being used as input to an older TimesTen release, or to some other database system, such as Oracle. For this release, `-c` prevents the `INLINE` and `NOT INLINE` keywords from being output.
DSN	Specifies an ODBC data source name of the database from which to get a schema.
`-fixedTypes`	Uses fully qualified data type names regardless of the current `TypeMode` value.
`-h` `-help -?`	Prints a usage message and exits.
`-l`	One per-line listing of objects in the database.
`-list {all | tables | views | sequences | cachegroups | repschemes | plsql}[,...]`	A comma-delimited (no space after comma) list of objects to generate. Lists only those types of objects specified. Default is `-list all`. `-list views` also displays information about materialized view logs.
`[`*owner*`.]`*object_name*	Limits the scope of the output to specified database object(s).
`-plsqlAttrs |-noplsqlAttrs`	Controls whether `ttSchema` emits `ALTER SESSION` statements with `CREATE` statements for PL/SQL program units. If `-plsqlAttrs` is specified, `ttSchema` emits `ALTER SESSION` statements to set these attributes before emitting a `CREATE` statement. This output from `ttSchema` can be fed back into `ttIsql` (or `sqlplus`) to create the same procedures, with the same compiler options as were specified in the original database. (default) If `-noplsqlAttrs` is specified, only the `CREATE` statement is generated.
`-plsqlCreate | -plsqlCreateOrReplace`	If `-plsqlCreate` is specified, `ttSchema` emits `CREATE PROCEDURE`, `CREATE PACKAGE` or `CREATE FUNCTION` statements for PL/SQL program units. If `-plsqlCreateOrReplace` is specified, `ttSchema` emits `CREATE` OR `REPLACE` statements. (default)
`-st | -systemTables`	Include system tables. System tables are omitted by default.
`-V | -version`	Prints the release number of `ttSchema` and exits.

Examples

Objects in the `orderdsn` database are created with these SQL statements:

```
CREATE TABLE ttuser.customer (
cust_num            INTEGER NOT NULL PRIMARY KEY,
  region            CHAR(2) NOT NULL,
  name              VARCHAR2(80),
  address           VARCHAR2(255) NOT NULL);

CREATE SEQUENCE ttuser.custid MINVALUE 1 MAXVALUE 1000000;

CREATE TABLE ttuser.orders (
  ord_num INTEGER NOT NULL PRIMARY KEY,
```

```
  cust_num INTEGER NOT NULL,
  when_placed  TIMESTAMP NOT NULL,
  when_shipped TIMESTAMP,
  FOREIGN KEY(cust_num) REFERENCES ttuser.customer (cust_num));

CREATE MATERIALIZED VIEW ttuser.order_summary AS
  SELECT cust.name, ord.ord_num, count(*) ord_count
  FROM ttuser.orders ord, ttuser.customer cust
  WHERE ord.cust_num = cust.cust_num
    GROUP BY cust.name, ord.ord_num;
```

Example 3–4 ttSchema for the database

Return the schema for the `orderdsn` database.

```
% ttSchema orderdsn
-- Database is in Oracle type mode
create table TTUSER.CUSTOMER (
        CUST_NUM NUMBER(38) NOT NULL,
        REGION   CHAR(2 BYTE) NOT NULL,
        "NAME"   VARCHAR2(80 BYTE) INLINE NOT NULL,
        ADDRESS  VARCHAR2(255 BYTE) NOT INLINE NOT NULL,
    primary key (CUST_NUM));

create table TTUSER.ORDERS (
        ORD_NUM      NUMBER(38) NOT NULL,
        CUST_NUM     NUMBER(38) NOT NULL,
        WHEN_PLACED  TIMESTAMP(6) NOT NULL,
        WHEN_SHIPPED TIMESTAMP(6),
    primary key (ORD_NUM),
    foreign key (CUST_NUM) references TTUSER.CUSTOMER (CUST_NUM));

create sequence TTUSER.CUSTID
    increment by 1
    minvalue 1
    maxvalue 1000000
    start with 1
    cache 20;

create materialized view TTUSER.ORDER_SUMMERY as
    SELECT CUST.NAME "NAME", ORD.ORD_NUM "ORD_NUM", COUNT(*) "ORD_COUNT"
    FROM TTUSER.ORDERS ORD, TTUSER.CUSTOMER CUST WHERE ORD.CUST_NUM =
    CUST.CUST_NUM GROUP BY CUST.NAME, ORD.ORD_NUM ;
```

Example 3–5 Listing sequences

Return the sequences for the `orderdsn` database.

```
% ttSchema -list sequences orderdsn
-- Database is in Oracle type mode
create sequence TTUSER.CUSTID
    increment by 1
    minvalue 1
    maxvalue 1000000
    start with 1
    cache 20;
```

Example 3–6 Specifying an object

Return the schema information for the `orders` table in the `orderdsn` database.

```
% ttSchema orderdsn ttuser.orders
-- Database is in Oracle type mode
Warning: tables may not be printed in an order that can satisfy foreign key
reference constraints
create table TTUSER.ORDERS (
        ORD_NUM      NUMBER(38) NOT NULL,
        CUST_NUM     NUMBER(38) NOT NULL,
        WHEN_PLACED  TIMESTAMP(6) NOT NULL,
        WHEN_SHIPPED TIMESTAMP(6),
    primary key (ORD_NUM),
    foreign key (CUST_NUM) references TTUSER.CUSTOMER (CUST_NUM));
```

Example 3–7 Specifying fixed data types

Return the schema information for the `orderdsn` database, using fixed data type names.

```
% ttSchema -fixedTypes orderdsn
-- Database is in Oracle type mode
create table TTUSER.CUSTOMER (
        CUST_NUM NUMBER(38) NOT NULL,
        REGION   ORA_CHAR(2 BYTE) NOT NULL,
        "NAME"   ORA_VARCHAR2(80 BYTE) INLINE NOT NULL,
        ADDRESS  ORA_VARCHAR2(255 BYTE) NOT INLINE NOT NULL,
    primary key (CUST_NUM));

create table TTUSER.ORDERS (
        ORD_NUM      NUMBER(38) NOT NULL,
        CUST_NUM     NUMBER(38) NOT NULL,
        WHEN_PLACED  ORA_TIMESTAMP(6) NOT NULL,
        WHEN_SHIPPED ORA_TIMESTAMP(6),
    primary key (ORD_NUM),
    foreign key (CUST_NUM) references TTUSER.CUSTOMER (CUST_NUM));

create sequence TTUSER.CUSTID
    increment by 1
    minvalue 1
    maxvalue 1000000
    start with 1
    cache 20;

create materialized view TTUSER.ORDER_SUMMERY as
    SELECT CUST.NAME "NAME", ORD.ORD_NUM "ORD_NUM", COUNT(*) "ORD_COUNT" FROM
    TTUSER.ORDERS ORD, TTUSER.CUSTOMER CUST WHERE ORD.CUST_NUM = CUST.CUST_NUM
    GROUP BY CUST.NAME, ORD.ORD_NUM ;
```

Notes

The SQL generated does not produce a history of transformations through `ALTER` statements, nor does it preserve table partitions, although the output gives information on table partitions in the form of SQL comments.

Output is not guaranteed to be compatible with DDL recognized by previous releases of TimesTen.

ttSize

Description

Estimates the amount of space that a given table, including any views in the database will consume when the table grows to include *rows* rows. It can be used on existing tables or to estimate table sizes when creating tables. If no owner is specified, the size information is printed for all tables of the given name. The size information includes space occupied by any indexes defined on the table.

The memory required for varying-length columns is estimated by using the average length of the columns in the current table as the average length of the columns in the final table. If there are no rows in the current table, then `ttSize` assumes that the average column length is one half the maximum column length.

The table is scanned when this utility is called. The scan of the table can be avoided by specifying an optional non-`NULL` *frac* value, which should be between `0` and `1`. This value is used to estimate the average size of varying-length columns. The maximum size of each varying-length column is multiplied by the *frac* value to compute the estimated average size of `VARBINARY` or `VARCHAR` columns. If the *frac* option is not specified, the existing rows in the table are scanned and the average length of the varying-length columns in the existing rows is used. If *frac* is not specified and the table has no rows in it, then *frac* is assumed to have the value 0.5.

Required privilege

This utility requires no privileges beyond those needed to perform select operations on the specified database objects.

Syntax

```
ttSize {-h | -help | -?}
ttSize {-V | -version}
ttSize -tbl [owner.]tableName [-rows rows] [- frac fraction]
     {-connStr connection_string | DSN}
```

Options

`ttSize` has the options:

Option	Description
`-connStr` *connection_string*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
DSN	Specifies the name of a data source to which `ttSize` should connect to retrieve table information.
`-frac` *fraction*	Estimated average fraction of `VARBINARY` or `VARCHAR` column sizes.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

Option	Description
`-rows` *`rows`*	Specifies the expected number of rows in the table. Space required to store a TimesTen table includes space for the actual data, plus overhead for bookkeeping, dynamic memory allocation and indexes. TimesTen may consume additional space due to memory fragmentation, temporary space allocated during query execution and space to hold compiled SQL statements. If this option is omitted, the number of rows in the existing table is used, or 1 if the table is empty.
`-tbl` [*`owner.`*]*`tableName`*	Specifies the name of the table whose definition should be used for size estimation. If the owner is omitted, the login name of the user is tried. If that is not found, the user `SYS` is used.
`-V` | `-version`	Prints the release number of `ttSize` and exits.

Example

To estimate the space required for a table, create the table in TimesTen, populate it with a sample of representative rows, create desired indexes and execute `ttSize` with those definitions. For example, to estimate the size of the `NAMEID` table in the data source `FixedDs` when it grows to 200,000 rows, execute:

```
ttSize FixedDs -tbl Nameid -rows 200000
```

Notes

Another method for estimating size requirements and measuring fragmentation is to use the MONITOR table. (See "SYS.MONITOR" in *Oracle TimesTen In-Memory Database System Tables and Limits Reference*.)

The columns `PERM_ALLOCATED_SIZE` and `PERM_IN_USE_SIZE` show the currently allocated size of the database (in KB units) and the in-use size of the database. The system updates this information each time a connection is made or released and each time a transaction is committed or rolled back.

This utility is supported only for TimesTen Data Manager DSNs. It is not supported for TimesTen Client DSNs.

See also

`ttSize`

ttStatus

Description

Displays information that describes the current state of TimesTen. The command displays:

- State of the TimesTen daemon process and all subdaemon processes.
- Names of all existing TimesTen databases.
- Number of users currently connected to each TimesTen database.
- The RAM, cache agent and replication policies.
- TimesTen cache agent status.
- The status of PL/SQL.
- The key and address of the shared memory segment used by TimesTen.
- The address, key and ID of the shared memory segment used by PL/SQL.
- Whether the TimesTen instance is accessible by a specified operating system group or accessible by anyone. For more details, see the daemon options in the "Managing TimesTen daemon options" in the *Oracle TimesTen In-Memory Database Operations Guide*.
- Miscellaneous status information.

Required privilege

This utility requires no privileges.

Syntax

```
ttStatus {-h | -help | -?}
ttStatus {-V | -version}
ttStatus [-v] [-r secs] [-[no]pretty]
```

Options

`ttStatus` has the options:

Option	Description

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-[no]pretty`	Do [not] use pretty formatting. The default is pretty formatting, which uses the values of the `ConnectionName` attribute.
`-r` *`secs`*	Allows `ttStatus` to continue running. Updates status report every secs seconds.
`-V` | `-version`	Prints the release number of `ttStatus` and exits.
`-v`	Prints detailed information that is useful for TimesTen customer support.

Sample output

When you invoke the command, a report that describes the current state of the system is displayed to `stdout`. The following is sample output:

```
TimesTen
 status report as of Thu Jul 17 19:15:41 2008

Daemon pid 26622 port 4133 instance sadrake
No TimesTen server running

----------------------------------------------------------------------
Data store /scratch/timesten_user/rdbms
There are 7 connections to the data store
Data store is in shared mode
Shared Memory KEY 0x13014429 ID 80347138
PL/SQL Memory KEY 0x14014429 ID 80379907 Address 0x2345678900000000
Type            PID     Context     Connection Name              ConnID
Process         26743   0x08117148  rdbms                             1
Subdaemon       26626   0x080bdfa8  Worker                         2042
Subdaemon       26626   0x08159228  Flusher                        2043
Subdaemon       26626   0x081c8850  Monitor                        2044
Subdaemon       26626   0x08217c10  HistGC                         2045
Subdaemon       26626   0x08266fd0  Aging                          2046
Subdaemon       26626   0x082b6390  Checkpoint                     2047
Replication policy  : Manual
Cache agent policy  : Manual
Access control enabled.
PL/SQL enabled.

End of report
```

When you invoke the command with the `-pretty` option, a report that describes the current state of the system is displayed to `stdout`. The following is sample output:

```
Data store /ds0/timesten_user/sample
          There are 7 connections to the data store
          Data store is in shared mode
          Shared Memory KEY 0x13014429 ID 80347138
          PL/SQL Memory KEY 0x14014429 ID 80379907 Address 0x2345678900000000
      Type            PID     Context             Connection Name      ConnID
Process        29508   0x00000001001c6680  rdbms                     1
Process         26743   0x08117148  rdbms                             1
Subdaemon       26626   0x080bdfa8  Worker                         2042
Subdaemon       26626   0x08159228  Flusher                        2043
Subdaemon       26626   0x081c8850  Monitor                        2044
Subdaemon       26626   0x08217c10  HistGC                         2045
Subdaemon       26626   0x08266fd0  Aging                          2046
Subdaemon       26626   0x082b6390  Checkpoint                     2047

          Replication policy  : Manual
          Cache agent policy : Manual
          Access control enabled.
          PL/SQL enabled.
```

Notes

While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

See also

`ttAdmin`

ttSyslogCheck (UNIX)

Description

Determines if the system's `/etc/syslog.conf` file is properly configured for TimesTen. The TimesTen Data Manager uses `syslog` to log a variety of progress messages. It is highly desirable to configure `syslog` so that all TimesTen messages are written to disk in a single disk file. The `ttSyslogCheck` utility examines the `syslog` configuration (in `/etc/syslog.conf`) to verify that it is properly configured for TimesTen.

If `syslog` is properly configured, `ttSyslogCheck` displays the name of the file that TimesTen messages are logged to and exits with exit code 0. If `syslog` is not properly configured, `ttSyslogCheck` displays an error message and exits with code 1.

Required privilege

This utility requires no privileges.

Syntax

```
ttSyslogCheck {-h | -help | -?}
ttSyslogCheck {-V | -version}
ttSyslogCheck [-facility name]
```

Options

`ttSyslogCheck` has the options:

Option	Description

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-facility` *name*	Specifies the `syslog` facility name being used for message logging.
`-V | -version`	Prints the release number of `ttSyslogCheck` and exits.

Note

This utility is supported only where the TimesTen Data Manager is installed.

ttTail

Description

Fetches TimesTen internal trace information from a database and displays it to `stdout`. By default, TimesTen generates no tracing information. See "ttTraceMon" on page 3-115 for more information.

Required privilege

This utility requires the ADMIN privilege.

Syntax

```
ttTail {-h | -help | -?}
ttTail {-V | -version}
ttTail [-f] {-connStr connection_string | DSN}
```

Options

The `ttTail` utility supports the options:

Option	Description
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Indicates the ODBC data source name of the database from which to get a trace.
`-f`	When the end of the trace is reached, `ttTail` does not terminate but continues to execute, periodically polling the database's trace buffer to retrieve and display additional TimesTen trace records. For example, this is useful for generating a display of trace data that is updated in real time.
`-h-help` `-?`	Prints a usage message and exits.
`-V | -version`	Prints the release number of `ttTail` and exits.

Example

```
ttTail MyDatastore
```

Notes

While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

ttTraceMon

Description

The `ttTraceMon` utility lets you enable and disable the TimesTen internal tracing facilities.

Tracing options can be enabled and disabled separately for each database. Each database contains a trace buffer into which messages describing TimesTen internal operations can be written. By default, tracing is disabled. However, it can be enabled using this utility.

The `ttTraceMon` utility provides subcommands to enable, disable, dump and manipulate trace information. `ttTraceMon` can be executed interactively (multiple subcommands can be entered at a prompt) or not interactively (one subcommand can be specified on the `ttTraceMon` command line).

When executed interactively, `ttTraceMon` prompts for lines of text from standard input and interprets the lines as trace commands. You can provide multiple trace commands on the same line by separating them with semicolons. To exit `ttTraceMon`, enter a blank line.

In interactive mode, you can redirect `ttTraceMon` command output to a file:

```
ttTraceMon connection_string >filename
```

Component names are case-insensitive. Some commands (`dump`, `show` and `flush`) allow you to list many components and operate on each one. For each subcommand, if you do not list components, the utility operates on all components.

For a description of the components available through this utility and a description of the information that `ttTraceMon` returns for each, see "Using the ttTraceMon utility" in the *Oracle TimesTen In-Memory Database Troubleshooting Procedures Guide*.

Required privilege

This utility requires the `ADMIN` privilege.

Syntax

```
ttTraceMon {-h | -help | -?}
ttTraceMon {-V | -version}
ttTraceMon [-e subcommand] {-connStr connection_string | DSN}
```

Options

`ttTraceMon` has the options:

Option	Description
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Indicates the ODBC data source name of the database from which to get trace information.

Option	Description
`-e` *`subcommand`*	Causes the subcommand to be executed against the specified database. If the subcommand consists of more than one word, enclose it in double quotes. For example: `ttTraceMon -e "show err" SalesData` Once the subcommand is complete, `ttTraceMon` exits. If `-e` is not specified, `ttTraceMon` starts in interactive mode, reading commands from `stdin` and displaying results to `stdout`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-V | -version`	Prints the release number of `ttTraceMon` and exits.

Subcommands

ttTraceMon can be called with the following subcommands:

Command	Description
`components`	List the names and internal identifiers of all `components`. For a description of the components available through this utility and a description of the information that `ttTraceMon` returns for each, see "Using the `ttTraceMon` utility" in the *Oracle TimesTen In-Memory Database Troubleshooting Procedures Guide*.
`connection {all | self |` *`connectionNum`*`} [on |off]`	Turn tracing on/off for specified connection. At database creation, tracing is "on" for all connections. The value for *`connectionNum`* is the connection slot number or the first number in the transaction ID.
`dump`	Prints all trace records currently buffered.Requires SELECT privileges or database object ownership.
`dump` *`comp`*	Prints all trace records for component *`comp`*.Requires SELECT privileges or database object ownership.
`flush`	Discards all buffered trace records.
`flush` *`comp`*	Discards all buffered trace records for component *`comp`*.
`help`	Prints a summary of the trace commands.
`level` *`comp n`*	Sets the trace level for component *`comp`* to *`n`*.Requires ADMIN privileges or database object ownership.
`outfile` *`file`*	Prints trace output to the specified file. The file may be any of 0, `stdout`, `stderr`, or a file name. On Windows, the file name must be in short 8.3 format. Printing is turned off when file is 0. TimesTen continues to buffer traces as usual, and they are accessible through other utilities like `ttTail`. If no *`file`* is specified, prints the current outfile setting.
`show`	Shows all the trace levels in force.
`show` *`comp`*	Shows the trace level for component *`comp`*.

Notes

Because tracing can degrade performance significantly, we recommend that you enable tracing only to debug problems. While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

ttuser

Description

Prompts for a password and returns an encrypted password. You can then include the output in a connection string or as the value for the `PWDCrypt` attribute in an ODBCINI file.

Required privilege

This utility requires no privileges.

Syntax

```
ttuser {-h | -help | -?}
ttuser {-V | -version}
ttuser -pwdcrypt
```

Options

The `ttuser` utility supports the options:

Option	Description

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-pwdcrypt`	Generates an encrypted password value for the `PWDCrypt` attribute.
`-V | -version`	Prints the release number of `ttuser` and exits.

ttVersion

Description

The `ttVersion` utility lists the TimesTen release information, including: number, platform, instance name, instance admin, instance home directory, daemon home directory, port number and build timestamp.

Required privilege

This utility requires no privileges.

Syntax

```
ttVersion [-m]
ttVersion [-m]
```

Options

`ttVersion` has the option:

Option	Description
`-m`	Generates computer-readable enhanced output. If not specified, abbreviated information is output.

Output

`ttVersion` produces the following sample output.

```
TimesTen Release 11.2.1 (32 bit Linux/x86) (tt1121_32:53384) 2009-05-26T23:00:04Z
Instance admin: terry
Instance home directory: spider/terry/TimesTen/tt1121_32
Daemon home directory: spider/terry/TimesTen/tt1121_32/srv/info
```

`ttVersion -m` produces the following sample output. Most of the entries only appear for patch releases.

```
patched=yes
product=TimesTen
major1=11
major2=2
major3=1
patch=1
portpatch=0
version=11.2.1.1.0
shortversion=1121
numversion=110200010100
bits=32
os=Linux/x86
buildtstamp=1243378804
buildtime=2009-05-26T23:00:04Z
clientonly=no
instance=tt1121_32
effective_port=53384
orig_port=53384
instance_admin=terry
effective_insthome=/spider/terry/ttcur/TTBuild/linux86_dbg/tt1121_32
orig_insthome=/spider/terry/ttcur/TTBuild/linux86_dbg/tt1121_32
```

```
effective_daemonhome=/spider/terry/ttcur/TTBuild/linux86_dbg/tt1121_32/info
orig_daemonhome=/spider/terry/ttcur/TTBuild/linux86_dbg/tt1121_32/info
plsql=0
groupname=timesten
```

ttXactAdmin

Description

The `ttXactAdmin` utility lists ownership, status, log and lock information for each outstanding transaction. The `ttXactAdmin` utility also allows you to heuristically commit, abort or forget an XA transaction branch.

Required privilege

This utility requires various privileges depending on which options are entered on the command line. See the description of the options to determine what privilege is needed, if any.

Syntax

```
ttXactAdmin {-h | -help | -?}
ttXactAdmin {-V | -version}
ttXactAdmin [-v verbosity] [-lsn] [-mt maxTrans] [-ml maxLocks] [-pid pid]
[-xact xid] [-tbl [owner.]tableName][-row rowid] [-interval seconds]
[-count iterations] {DSN | -connstr connectionString}
ttXactAdmin -latch [-interval seconds] [-count iterations]
{DSN | -connstr connStr}
ttXactAdmin -latchstats clear | off |on | [show] [-interval seconds]
[-count iterations] {DSN | -connstr connectionString}
ttXactAdmin -connections [-pid pid] [-interval seconds]
[-count iterations] {DSN | -connstr connStr}
ttXactAdmin -xactIdRollback xid {DSN | -connstr connStr}
ttXactAdmin {-HCommit xid | -HAbort xid | -HForget xid} {DSN | -connstr connStr}
```

Options

`ttXactAdmin` has the options:

Option	Description
`-connections`	Shows all current connections to the database. When run with the `-connections` option, `ttXactAdmin` itself does not establish a true connection to the database, and requires no latches. This can be useful when diagnosing frozen systems. This option requires `ADMIN` privileges.
`-connStr` *`connectionString`*	An `ODBC` connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
`-count` *`iterations`*	Generate the report iterations times. If no `-interval` option is specified, an interval of 1 second is used.
`DSN`	Indicates the ODBC data source name of the database to be administered.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-HAbort` *`xid`*	Heuristically abort an XA transaction branch in TimesTen. The specified transaction ID must be the local TimesTen TransID. This option requires ADMIN privileges or ownership of the specified transactions.

Option	Description
`-HCommit` *`xid`*	Heuristically commit an XA transaction branch in TimesTen. The specified transaction ID must be the local TimesTen TransID. This option requires ADMIN privileges or ownership of the specified transactions.
`-HForget` *`xid`*	Heuristically forget an XA transaction branch in TimesTen. The specified transaction ID must be the local TimesTen TransID. This option requires ADMIN privileges or ownership of the specified transactions.
`-interval` *`seconds`*	Repeat the generation of the report, pausing the indicated number of seconds between each generation. If no `-count` option is specified, repeat forever.
`-latch`	This option is to be used by TimesTen Customer Support only. Shows only the latch information for the database specified.
`-latchstats[clear | off | on | show]`	This option is to be used by TimesTen Customer Support only. Performs the requested latchstat operation. This option requires ADMIN privileges. All other options are ignored when `-latchstats` is used. `clear` - Resets all latchstat information to zero. `off` - Turns off collection of latchstats. `on` - Turns on collection of latchstats. `show` - Shows the latch information, including access counts and other stats. Default, if no operation is specified.
`-ml` *`maxLocks`*	Maximum number of locks per transaction. Default is 6000.
`-mt` *`maxTrans`*	Specifies the maximum number of transactions to be displayed. The default is all outstanding transactions.
`-pid` *`pid`*	Displays only transactions started by the process with the specified pid. On Linux, it is the pid of the thread that opens the connection. This option requires ADMIN privileges or ownership of the specified transactions.
`-row` *`rowid`*	Displays lock information for the specified row. This option requires ADMIN privileges or database object ownership.
`-tbl [`*`owner`*`.]`*`tableName`*	Displays lock information for the specified table. This option requires ADMIN privileges or ownership of the specified table.
`-V | -version`	Prints the release number of `ttXactAdmin` and exits.
`-v` *`verbosity`*	Specifies the verbosity level. One of: `0` – does not display the names of the tables for row locks. In this case, `ttXactAdmin` runs faster.`1`-(the default) displays the names of the tables for row locks.

Option	Description
`-xact` *`xid`*	Displays information for the specified transaction, including its log hold LSN. In the output, the field "Last ID" is a set of two sequence numbers. If the sequence numbers did not change in an interval, then no log record was written by the transaction during that interval. This option requires `ADMIN` privileges or ownership of the specified transactions.
`-xactIdRollback` *`xid`*	Allows you to roll back a transaction. This may be particularly useful for long running transactions. This parameter xid represents the transaction ID.This stops any currently executing operations on behalf of that transaction and then rolls back the transaction in TimesTen. If there is currently a checkpoint in process when the rollback is requested, TimesTen terminates the checkpoint operation.This command does not stop IMDB Cache operations on Oracle. Operations include passthrough statements, flushing, manual loading, manual refreshing, synchronous writethrough, propagating, and transparent loading. This option requires ADMIN privileges or ownership of the specified transactions.

Output

ttXactAdmin produces the following output:

Column	Description
`Program File Name`	The executable file name of the process that owns the transaction
`PID`	The process ID of the application that owns the transaction. On Linux, the PID of the thread that opens the connection.
`Context`	The internal identifier that distinguishes between multiple connections to the database made by a single multithreaded process.
`TransId`	The unique identifier for the transaction used internally by TimesTen. The identifier has two parts. The first part is a relatively small value (less than 2048), used to discriminate between transactions that are active at the same time. The second part is a potentially large value (an unsigned integer), and is used to discriminate between successive uses of the same first part. (The value wraps around if necessary.) Thus, identifiers 4.100 and 4.200 cannot be present at the same time. If 4.100 is seen, and then 4.200, this indicates that transaction 4.100 has completed (committed or rolled back).

Column	Description
TransStatus	Current status of the transaction, one of: ■ `Active` - Active transaction ■ `Aborting` - A transaction is in the process of aborting. See Notes for more information. ■ `Committing` - Committing transaction, locks are being released. ■ `Ckpointing` - A transaction doing checkpoint. ■ `Rep-Wait-Return` - Replicated transaction waiting Return Receipt/Commit. ■ `Idle` - A transaction branch currently not accessing data. ■ `Prepared` - Prepared transaction branch. ■ `Heur-Committed` - Heuristically committed transaction branch. ■ `Heur-Aborted` - Heuristically aborted transaction branch. ■ `Propagating` - TimesTen transaction waiting for Oracle to commit.
Resource	The type of the lock being requested: ■ `Row` - Row-level lock. ■ `HashedKey` - A lock held on a key value of a hash index; acquired when an operation requires a hash index to be updated. ■ `Table` - Table-level lock. ■ `EndScan` - End of table or range scan lock. ■ `Database` - Database-level lock. ■ `Command` - Command lock. ■ `Prepare` - Lock acquired while preparing commands. ■ `GrpComm` - Group commit lock. ■ `ReplHold` - Lock for replication hold. ■ `XlaHold` - Lock for XLA hold.
ResourceId	A unique identifier of each unique resource. The identifier is displayed in hexadecimal format with a few exception. `Table` and `CompCmd` are shown as decimal values. Row locks are shown in the ROWID character format.

Column	Description
Mode	A value used to determine the level of concurrency that the lock provides: S - Shared lock in serializable isolation. Sn - Shared lock in non-serializable isolation .U - Update lock in serializable isolation. Un - Update lock in non-serializable isolation. En - End-of-scan lock for non-serializable isolation. IRC - Intention shared lock in non-serializable isolation. IS - Intention shared lock in serializable isolation IU - Intention update lock in serializable isolation IUn - Intention update lock in non-serializable isolationIX - Intention exclusive lock in serializable isolation IXn - Intention exclusive lock non-serializable isolation SIX - Shared lock with intent to set an exclusive lock in serializable isolation. SIXn - Shared lock with intent to set an exclusive lock non-serializable isolation. X - Exclusive lock. Xn - Exclusive lock in non-serializable isolation. W - Update, insert or delete table lock. XNi - Next lock for inserting into tables or non-unique index NS - Table lock in read-committed isolation that conflicts with all table locks in serializable isolationLock "0" means the blocker is still in the waiting list.
HMode	The mode in which the competing transaction is holding the lock which the waiting transaction is requesting. See "Mode" in this table for concurrency level descriptions.
RMode	Shows the mode in which the waiting transaction has requested to hold the lock. See "Mode" in this table for concurrency level descriptions.
HolderTransId	The identifier of the transaction with which the waiting transaction is in contention.
Name	The name of the table that the lock is being held on or within.

Example

The following command displays all locks in the database:

```
ttXactAdmin -connstr DSN=demodata

Outstanding locks
PID   Context  TransId     TransStatus Resource  ResourceId         Mode Name
Program File Name: localtest
10546 0x118e28 2047.000003 Active      Table     411104             IS  SYS.TABLES
                                       Table     416480             IXn TEST1.TAB1
                                       Row       BMUFVUAAABQAAAAGTD Sn  SYS.TABLES
                                       Hashed Key 0x69cf9c36        Sn  SYS.TABLES
                                       Database   0x01312d00        IX
                                       Row       BMUFVUAAABQAAAAGzD Xn  TEST1.TAB1
Program File Name: /users/smith/demo/XAtest1
XA-XID: 0xbea1-001b238716dc35a7425-64280531947e1657380c5b8d
```

```
1817 0x118e28 2046.000004 Active       Table    416480              IS  TEST1.TAB1
                                       CompCmd  21662408            S
                                       Database 20000000            IS
                                       Row      BMUFVUAAABQAAAAJzD  Sn  TEST1.TAB1
Program File Name: /users/smith/demo/XAtest2
XA-XID: 0xbea1-001c99476cf9b21e85e1-70657473746f7265506f6f6c
27317 0x118e28 2045.000005 Prepared    Table  411104             IS   SYS.TABLES
                                       Table  416816             IXn  TEST1.TAB2
                                       Row    BMUFVUAAABQAAAAMzD Sn   SYS.TABLES
                                       Database   0x01312d00     IX
                                       Hashed Key 0x67fe3852     Sn   SYS.TABLES
                                       Row    BMUFVUAAABQAAAAHTE Xn   TEST1.TAB2
Program File Name: /users/smith/demo/Reptest
27589 0x118e28 2044.000006 Rep-Wait-Return
Awaiting locks
PID  Context  TransId     Resource ResourceId RMode HolderTransId HMode Name
Program File Name: /users/smith/demo/XAtest1
1817 0x118e28 2046.000004 Row BMUFVUAAABQAAAAPTD  Sn  2047.000003   Xn  TEST1.TAB1
```

The following command displays all locks for transaction 2045.000005:

```
ttXactAdmin -xact 2045.000005 -connstr DSN=demodata
PID   Context  TransStatus 1stLSN    LastLSN  Resource ResourceId Mode Name
Program File Name: /users/smith/demo/XAtest2
XA-XID: 0xbea1-001c99476cf9b21e85e1-70657473746f7265506f6f6c
27317 0x118e28 Prepared    0.0116404 0.0116452 Table 411104         IS  SYS.TABLES
                                               Table   416816         IXn TEST1.TAB2
                                               Row BMUFVUAAABQAAAAGzE Sn  SYS.TABLES
                                               Database 0x01312d00    IXn
                                               Hashed Key 0x67fe3852  Sn SYS.TABLES
                                               Row BMUFVUAAABQAAAAKzE Xn TEST1.TAB2
```

To display all the connections to the database:

```
$ ttXactAdmin -connections sample
2006-09-10 10:26:33
/datastore/terry/sample
TimesTen Release 11.2.1.0.0

ID   PID    Context            Name           Program        State TransID UID
1    29508  0x00000001001c6680 myconnection   ttIsql         Run   1.23    TERRY
2044 29505  0x0000000100165290 Worker         timestensubd   Run           TERRY
2045 29505  0x00000001001df190 Flusher        timestensubd   Run           TERRY
2046 29505  0x000000010021cc50 Monitor        timestensubd   Run           TERRY
2047 29505  0x0000000100206730 Checkpoint     timestensubd   Run           TERRY
    5 connections found
```

Notes

If the transaction specified in the command is not an XA transaction branch but a TimesTen local transaction, no XA-XID are displayed. The XA-XID is a C structure that contains a format identifier, two length fields and a data field. The data field consists of at most two contiguous components: a global transaction identifier (*gtrid*) and a branch qualifier (*bqual*). The two length fields specify the number of bytes (1-64) in *gtrid* and *bqual* respectively. For more details, refer to the *X/Open publication: Distributed Transaction Processing: The XA Specification (c193)*.

Under RMode, awaiting transactions are sorted by PID and Context. The listing does not reflect the order of the lock requests.

A lock request with an RMode compatible with the HMode of the lock holder can be waiting because there is another lock request with an incompatible mode ahead of the compatible request in the lock request queue.

A transaction can have the status `Aborting` for one of these reasons:

- A user application requested rollback after doing a large amount of work.
- An application with autocommit tried a statement that could not be completed and it is being undone.
- Another call to `ttXactAdmin` caused a transaction to rollback.
- A process died with work in progress and that work is being undone.

ttXactLog

Description

Displays a formatted dump of the contents of a TimesTen transaction log. It is designed to be used by TimesTen customer support to diagnose problems in the log or database. A loss of data can occur with certain options such as -tr, therefore only use this tool if you have been asked to do so by a TimesTen customer support representative.

Required privilege

This utility requires the ADMIN privilege.

Syntax

```
ttXactLog {-h | -help | -?}
ttXactLog {-V | -version}
ttXactLog [-v verbosity] [-m maxChars] [-s] [-t] [-b blkID]
[-l1 lfn.lfo [-l2 lfn.lfo]] [-r recType] [...] [-tr dir]
[-lb] [-headers recs] [-logdir dir]
{-connStr connection_string | DSN | dspath}
```

Options

`ttXactLog` has the options:

Option	Description
`-b blkID`	Restricts log records to those accessing this block, plus any transaction records.
`-connStr connectionString`	An ODBC connection string containing the name of the database, the server name and DSN (if necessary) and any relevant connection attributes.
DSN	The ODBC source name of the database for which to display the transaction log.
dspath	The fully qualified name of the database. This is not the DSN associated with the connection but the fully qualified database path name associated with the database as specified in the `DataStore=` parameter of the database's ODBC definition. For example, for a database consisting of files`/home/payroll/1997.ds0, /home/payroll/1997.ds1` and several transaction log files `/home/payroll/1997.logn`,dspath is `/home/payroll/1997`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-headers records`	Prints one header for every *records* records. A value of 0 disables headers entirely.
`-lb`	Connects to the database and prints out the log buffer. Contents of the transaction log files are not printed. Requires SELECT privileges or database object ownership.
lfn.lfo	Transaction log file number (*lfn*) and transaction log file offset (*lfo*) for a log record.

Option	Description
`-l1`	Considers this log record only (unless an `-l2` argument is present).
`-l2`	Considers records between `-l1` and `-l2`, inclusive.
`-logdir` *`dir`*	Specifies the directory where the database's transaction log files reside. If `-logdir` is not specified, `ttXactLog` uses the directory path portion of the value supplied in dspath.
`-m` *`maxChars`*	Maximum number of characters printed for binary items (for `-v 3`) only (defaults to 1000).
`-r` *`recType`*	Considers only records of the specified type. This option may be used multiple times to specify a list of desired log record types. recType is case-sensitive.
`-s`	Prints summary information. Requires SELECT privileges or database object ownership.
`-t`	Only reads transaction log file tail (from start of last checkpoint transaction log file or, if no checkpoint, the most recent transaction log file).
`-tr` *`dir`*	Truncates all log records in the directory at the `LWN` boundary. The original transaction log files are moved to the directory *`dir`*.
`-V | -version`	Prints the release number of `ttXactLog` and exits.
`-v` *`verbosity`*	Specifies the verbosity level. One of: 0-print only summary log information (if `-s` specified). 1-(the default) print log record headers too. 2-print log record bodies too, except long data. 3-print full log records (see `-m` option).

Example

```
ttXactLog -v 3 -m 100 /users/pat/TimesTen/Daily/F112697SS
```

Note

This utility is supported only where the TimesTen Data Manager is installed.

Index

A

Access Control
- Authenticate, 1-95
- DSN attributes, 1-2
- ttAgingLRUConfig, 2-2
- ttAgingScheduleNow, 2-5
- ttApplicationContext, 2-7
- ttBackupStatus, 2-8
- ttBookmark, 2-11
- ttCacheAutorefreshStatsGet, 2-12, 2-14
- ttCacheAwtThresholdSet, 2-21
- ttCacheConfig, 2-22
- ttCacheDBCgStatus, 2-26
- ttCachePolicyGet, 2-29
- ttCachePolicySet, 2-31
- ttCachePropagateFlagSet, 2-33
- ttCacheSqlGet, 2-34
- ttCacheStart, 2-36
- ttCacheStop, 2-37
- ttCacheUidGet, 2-38
- ttCacheUidPwdSet, 2-39
- ttCkpt, 2-40
- ttCkptBlocking, 2-42
- ttCompact, 2-51
- ttCompactTS, 2-52
- ttConfiguration, 2-53
- ttContext, 2-54
- ttDataStoreStatus, 2-55
- ttDurableCommit, 2-57
- ttGridAttach, 2-58, 2-64
- ttGridCheckOwner, 2-60
- ttGridCreate, 2-61
- ttGridDestroy, 2-62
- ttGridDetach, 2-63
- ttGridDetachAll, 2-64, 2-66
- ttGridDetachList, 2-65
- ttGridGlobalCGResume, 2-66
- ttGridGlobalCGSuspend, 2-67
- ttGridInfo, 2-68
- ttGridNameSet, 2-70
- ttGridNodeStatus, 2-71
- ttHostGridCheckOwner, 2-60
- ttHostNameGet, 2-73
- ttHostNameSet, 2-74
- ttLockLevel, 2-75
- ttLockWait, 2-76
- ttLogHolds, 2-78
- ttMonitorHighWaterReset, 2-80
- ttOptClearStats, 2-81
- ttOptEstimateStats, 2-83
- ttOptGetColStats, 2-85
- ttOptGetFlag, 2-86
- ttOptGetMaxCmdFreeListCnt, 2-87
- ttOptGetOrder, 2-88
- ttOptJoinOrder, 2-104
- ttOptSetColIntvlStats, 2-89
- ttOptSetColStats, 2-92
- ttOptSetFlag, 2-94
- ttOptSetMaxCmdFreeListCnt, 2-98
- ttOptSetMaxPriCmdFreeListCnt, 2-99
- ttOptSetOrder, 2-100
- ttOptSetTblStats, 2-102
- ttOptUpdateStats, 2-106
- ttOptUseIndex, 2-109
- ttPLSQLMemoryStats, 2-111
- ttRamPolicyGet, 2-113
- ttRamPolicySet, 2-115
- ttRedundantIndexCheck, 2-117
- ttRepDeactivate, 2-119
- ttReplicationStatus, 2-120
- ttRepPolicyGet, 2-122
- ttRepPolicySet, 2-124
- ttRepQueryThresholdGet, 2-126
- ttRepQueryThresholdSet, 2-127
- ttRepStart, 2-129
- ttRepStateGet, 2-130
- ttRepStateSave, 2-133
- ttRepStateSet, 2-135
- ttRepStop, 2-137
- ttRepSubscriberStateSet, 2-138
- ttRepSubscriberSync, 2-140
- ttRepSubscriberWait, 2-142
- ttRepSyncGet, 2-144
- ttRepSyncSet, 2-147
- ttRepSyncSubscriberStatus, 2-150
- ttRepTransmitGet, 2-151
- ttRepXactStatus, 2-154
- ttRepXactTokenGet, 2-156
- ttSetUserColumnID, 2-158
- ttSetUserTableID, 2-159
- ttSize, 2-160

ttSQLCmdCacheInfo, 2-162
ttSQLCmdQueryPlan, 2-165
ttUserPrivileges, 2-164
ttWarnOnLowMemory, 2-169
ttXactIdGet, 2-170
ttXlaBookmarkCreate, 2-171
ttXlaBookmarkDelete, 2-172
ttXlaSubscribe, 2-173
ttXlaUnsubscribe, 2-174
utilities, 3-1
access modes, conflicting, 1-65
AllFlags optimizer flag, 2-96
allocating disk space
first connection attributes
Preallocate, 1-17, 1-42
altering plan generation, 2-109, 2-111
ASCII files and ttBulkCp, 3-14
attributes
access control, 1-2
AutoCreate, 1-23
CacheAWTMethod, 1-87
CacheGridEnable, 1-89
CacheGridMsgWait, 1-90
CkptFrequency, 1-24
CkptLogVolume, 1-26
CkptRate, 1-27
ConnectionCharacterSet, 1-67
ConnectionName, 1-45
Connections, 1-28
Data Source Name, 1-10
DatabaseCharacterSet, 1-12
DataStore, 1-11
DDLCommitBehavior, 1-46
Description, 1-15
Diagnostics, 1-49
DuplicateBindMode, 1-50
DurableCommits, 1-51
DynamicLoadEnable, 1-92
DynamicLoadErrorMode, 1-93
ForceConnect, 1-29
Isolation, 1-52
LockLevel, 1-54
LockWait, 1-55
LogAutoTruncate, 1-30
LogBuffMB, 1-31
LogBufParallelism, 1-32
LogDir, 1-16
LogFileSize, 1-33
LogFlushMethod, 1-34
Logging, 1-35
LogPurge, 1-36
MatchLogOpts, 1-56
MaxConnsPerServer, 1-112
MemoryLock, 1-37
NLS_LENGTH_SEMANTICS, 1-68
NLS_NCHAR_CONV_EXCP, 1-69
NLS_SORT, 1-70
OracleNetServiceName, 1-94
OraclePWD, 1-95
Overwrite, 1-39
PassThrough, 1-96
PermSize, 1-40
PermWarnThreshold, 1-57
PLSCOPE_SETTINGS, 1-81
PLSQL, 1-75
PLSQL_CCFLAGS, 1-82
PLSQL_MEMORY_ADDRESS, 1-77
PLSQL_MEMORY_SIZE, 1-79
PLSQL_OPTIMIZE_LEVEL, 1-84
PLSQL_TIMEOUT, 1-85
Preallocate, 1-17
PrivateCommands, 1-58
PWD, 1-64
PWDCrypt, 1-59
QueryThreshold, 1-60
RACCallback, 1-100
ReceiverThreads, 1-41
RecoveryThreads, 1-42
ReplicationApplyOrdering, 1-18
ReplicationParallelism, 1-19
ReplicationTrack, 1-61
ServersPerDSN, 1-113
ServerStackSize, 1-114
SQLQueryTimeout, 1-62
TCP_Port, 1-102
TCP_Port2, 1-103
Temporary, 1-20
TempSize, 1-43
TempWarnThreshold, 1-63
TTC_FailoverPortRange, 1-104
TTC_Server, 1-105
TTC_Server_DSN, 1-107
TTC_Server_DSN2, 1-108
TTC_Server2, 1-106
TTC_Timeout, 1-109
TypeMode, 1-21
UID, 1-64
WaitForConnect, 1-65
autocommit, 2-148
AutoCreate attribute, 1-23

B

backups, 3-11
built-in procedures
ttAgingLRUConfig, 2-2
ttAgingScheduleNow, 2-5
ttApplicationContex, 2-7
ttApplicationContext, 2-7
ttBookmark, 2-11
ttCacheAgingIntervalSet, 2-21
ttCacheAWTMonitorConfig, 2-18
ttCacheAwtThresholdGet, 2-20
ttCacheConfig, 2-22
ttCacheDbCgStatus, 2-26
ttCachePolicyGet, 2-29
ttCachePolicySet, 2-31
ttCachePropagateFlagSet, 2-33
ttCacheSqlGet, 2-34
ttCacheStart, 2-36

ttCacheStop, 2-37
ttCacheUidPwdSet, 2-39
ttCkpt, 2-40
ttCkptBlocking, 2-42
ttCkptConfig, 2-44
ttCkptHistory, 2-47
ttCompact, 2-51
ttCompactTS, 2-52
ttConfiguration, 2-53
ttContext, 2-54
ttDataStoreStatus, 2-55
ttDurableCommit, 2-57
ttGridAttach, 2-58, 2-66
ttGridCheckOwner, 2-60
ttGridCreate, 2-61
ttGridDestroy, 2-62
ttGridDetach, 2-63
ttGridDetachAll, 2-64
ttGridGlobalCGSuspend, 2-67
ttGridInfo, 2-68
ttGridNameSet, 2-70
ttGridNodeStatus, 2-71
ttHostNameGet, 2-73
ttHostNameSet, 2-74
ttLockLevel, 2-75
ttLockWait, 2-76
ttLogHolds, 2-78
ttOptClearStats, 2-81
ttOptEstimateStats, 2-83
ttOptGetColStats, 2-85
ttOptGetFlag, 2-86
ttOptGetMaxCmdFreeListCnt, 2-87
ttOptGetOrder, 2-88
ttOptSetFlag, 2-94
ttOptSetMaxCmdFreeListCnt, 2-98
ttOptSetMaxPriCmdFreeListCnt, 2-99
ttOptSetOrder, 2-100
ttOptSetTblStats, 2-102
ttOptShowJoinOrder, 2-104
ttOptUpdateStats, 2-106
ttOptUseIndex, 2-109
ttPLSQLMemoryStats, 2-111
ttRamPolicyGet, 2-113
ttRamPolicySet, 2-115
ttRedundantIndexCheck, 2-117
ttRepDeactivate, 2-119
ttReplicationStatus, 2-120
ttRepPolicyGet, 2-122
ttRepPolicySet, 2-124
ttRepQueryThresholdGet, 2-126
ttRepQueryThresholdSet, 2-127
ttRepStart, 2-129
ttRepStateSet, 2-135
ttRepStop, 2-137
ttRepSubscriberStateSet, 2-138
ttRepSubscriberStatus, 2-150
ttRepSubscriberWait, 2-140, 2-142
ttRepSyncGet, 2-144
ttRepSyncSet, 2-147
ttRepTransmitGet, 2-151
ttRepTransmitSet, 2-152
ttRepXactStatus, 2-154
ttRepXactTokenGet, 2-156
ttSetUserColumnID, 2-158
ttSetUserTableID, 2-159
ttSize, 2-160
ttVersion, 2-168
ttWarnOnLowMemory, 2-169
ttXactIdGet transaction ID
displaying, 2-170
ttXlaBookmarkCreate, 2-171
ttXlaBookmarkDelete, 2-172
ttXlaSubscribe, 2-173
bulk copy, 3-14

C

cache agent
ttCacheConfig procedure, 2-22
ttCacheDbCgStatus procedure, 2-26
ttCachePolicyGet procedure, 2-29
ttCachePolicySet procedure, 2-31
CacheAWTMethod attribute, 1-87
CacheGridEnable attribute, 1-89
CacheGridMsgWait attribute, 1-90
caching Oracle data
ttCacheAgingIntervalSet procedure, 2-21
ttCacheConfig procedure, 2-22
ttCacheDbCgStatus procedure, 2-26
ttCachePolicyGet procedure, 2-29
ttCachePolicySet procedure, 2-31
ttCachePropagateFlagSet procedure, 2-33
ttCacheSqlGet procedure, 2-34
ttCacheStart procedure, 2-36
ttCacheStop procedure, 2-37
ttCacheUidPwdSet procedure, 2-39
CallableStatement
example, 2-1
changing lock level, 2-75
checkpoints, 2-40
ttCkpt, 2-40
ttCkptBlocking, 2-42
ttCkptConfig, 2-44
ttCkptHistory, 2-47
CkptFrequency attribute, 1-24
CkptLogVolume attribute, 1-26
CkptRate attribute, 1-27
clearing statistics, 2-81
Client connection attributes
TCP_Port, 1-102
TCP_Port2, 1-103
TTC_FailoverPortRange, 1-104
TTC_Server, 1-105
TTC_Server_DSN, 1-107
TTC_Server_DSN2, 1-108
TTC_Server2, 1-106
TTC_Timeout, 1-109
UID/PWD, 1-64
COL_STATS system table
updating explicitly, 2-106

compacting database, 2-51
concurrency, 1-54
conflicting access modes, 1-65
connection attribute
  DynamicLoadEnable, 1-92
connection failure and MatchLogOpts, 1-56
ConnectionCharacterSet attribute, 1-67
ConnectionName attribute, 1-45
Connections attribute, 1-28
consistency checking, 3-29
Correlation Name
  definition, 2-100
creating databases
  AutoCreate attribute, 1-23
  Overwrite attribute, 1-39
  PermSize attribute, 1-40
  Preallocate attribute, 1-17
  RecoveryThreads attribute, 1-42
  TempSize attribute, 1-43

D

Data Source Name attribute, 1-10
data store
  attributes, 1-9
  Data Source Name attribute, 1-10
  DatabaseCharacterSet attribute, 1-12
  DataStore attribute, 1-11
  Description attribute, 1-15
  LogDir attribute, 1-16
  Preallocate attribute, 1-17
  ReplicationApplyOrdering attribute, 1-18
  ReplicationParallelism, 1-19
  Temporary attribute, 1-20
  TypeMode, 1-21
database
  AutoCreate attribute, 1-23
  backups, 3-11
  calling ttCompact, 2-51
  calling ttCompactTS, 2-52
  compacting, 2-51
  creating, 1-23
  database-level locking, 1-54
  destroying, 3-45
  estimating size, 2-160, 3-108
  ForceConnect attribute, 1-29
  LockLevel attribute, 1-54
  LogAutoTruncate attribute, 1-30
  LogFlushMethod attribute, 1-34
  LogPurge attribute, 1-36
  MatchLogOpts attribute, 1-56
  Overwrite attribute, 1-39
  PermSize attribute, 1-40
  PermWarnThreshold attribute, 1-57
  PrivateCommands attribute, 1-58
  PWDCrypt attribute, 1-59
  QueryThreshold attribute, 1-60
  ReplicationApplyOrdering attribute, 1-18
  restoring, 3-102
  status, 2-55
  TempSize attribute, 1-43
  TempWarnThreshold attribute, 1-63
  WaitForConnect attribute, 1-65
DatabaseCharacterSet attribute, 1-12
database-level locking
  ttLockLevel procedure, 2-75
DataStore attribute, 1-11
data-store-level locking
  attribute, 1-54
  different levels of locking, 1-54
DDLCommitBehavior attribute, 1-46
Default optimizer flag, 2-96
deleting database, 3-45
Description attribute, 1-15
destroying database, 3-45
Diagnostics attribute, 1-49
displaying the TimesTen Version, 3-119
displaying transaction ID, 2-170
displaying transaction log, 3-128
DuplicateBindMode attribute, 1-50
durable commits
  ttDurableCommit procedure, 2-57
DurableCommits attribute, 1-51
DynamicLoadEnable connection attribute, 1-92
DynamicLoadEnable optimizer flag, 2-94
DynamicLoadErrorMode attribute, 1-93
DynamicLoadErrorMode optimizer flag, 2-94

E

eliminating fragmentation, 2-51
estimating statistics, 2-83
estimating table size, 2-160, 3-108

F

Firs tRow optimizer flag, 2-94
first connection attributes
  AutoCreate, 1-23
  LogPurge, 1-36
  Overwrite, 1-39
  PermSize, 1-40
  TempSize, 1-43
ForceConnect attribute, 1-29
fragmentation
  eliminating with ttCompact, 2-51
  eliminating with ttCompactTS, 2-52

G

general connection attributes
  ConnectionCharacterSet, 1-67
  ConnectionName, 1-45
  description, 1-44
  Diagnostics, 1-49
  DurableCommits, 1-51
  LockLevel, 1-54
  LockWait, 1-55
  MatchLogOpts, 1-56
  PLSQL, 1-75
  WaitForConnect, 1-65

GenPlan optimizer flag, 2-95
getting statistics, 2-85

H

Hash optimizer flag, 2-95

I

interactively running SQL, 3-47
internal consistency checking, 3-29
Isolation attribute, 1-52

L

locked database, 1-54
locking
  ttLockLevel procedure, 2-75
LockLevel attribute, 1-54
locks
  changing level, 2-75
  ttLockLevel procedure, 2-75
LockWait attribute, 1-55
log buffer size
  and logging to disk, 1-31
  attribute, 1-31
log file size
  and logging, 1-33
  attribute, 1-33
log files
  purging, 1-36
LogAutoTruncate attribute, 1-30
LogBuffSize attribute, 1-31, 1-32
LogBufParallelism attribute, 1-32
LogDir attribute, 1-16
LogFileSize attribute, 1-33
LogFlushMethod attribute, 1-34
Logging attribute, 1-35
  and MatchLogOpts, 1-56
LogPurge attribute, 1-36
  and MatchLogOpts, 1-56
  description, 1-36
logs
  transaction log, 3-128

M

making nondurable commits durable, 2-57
MatchLogOpts attribute, 1-56
MaxConnsPerServer attribute, 1-112
maximum
  client/server timeout period, 1-8
memory
  usage policy, 3-5
MemoryLock attribute, 1-37
migrating databases
  ttMigrate utility, 3-72
modifying column statistics explicitly, 2-89, 2-92
modifying table statistics explicitly, 2-102

N

NestedLoop optimizer flag, 2-95
NLS general connection attributes, 1-66
NLS_LENGTH_SEMANTICS attribute, 1-68
NLS_NCHAR_CONV_EXCP attribute, 1-69
NLS_SORT attribute, 1-70
nondurable commits, making durable, 2-57

O

optimizer flag
  settings, 2-86
  ttOptGetFlag procedure, 2-86
optimizer flags
  AllFlags, 2-96
  Default, 2-96
  FirstRow, 2-94
  GenPlan, 2-95
  Hash, 2-95
  NestedLoop, 2-95
  PassThrough, 2-95
  Range, 2-96
  Rowid, 2-96
  RowLock, 2-95, 2-97
  Scan, 2-95
  TblLock, 2-96, 2-97
  TmpHash, 2-96
  TmpRange, 2-96
  TmpTable, 2-96
Oracle Agent
  ttCacheStart procedure, 2-36
  ttCacheStop procedure, 2-37
Oracle Connect
  ttCacheAgingIntervalSet procedure, 2-21
  ttCachePropagateFlagSet procedure, 2-33
  ttCacheSqlGet procedure, 2-34
  ttCacheUidPwdSet procedure, 2-39
OracleID attribute, 1-94
OraclePWD attribute, 1-95
out-of memory warnings, 1-57, 1-63
Overwrite attribute, 1-39

P

parallel replication, 1-19
PassThrough attribute, 1-96
PassThrough optimizer flag, 2-95
PermSize attribute, 1-40
PermWarnThreshold attribute, 1-57
plan generation
  altering, 2-109, 2-111
PLSCOPE_SETTINGS attribute, 1-81
PLSQL attribute, 1-75
PL/SQL first connection attributes, 1-74
PLSQL_CCFLAGS attribute, 1-82
PLSQL_MEMORY_ADDRESS attribute, 1-77
PLSQL_MEMORY_SIZE attribute, 1-79
PLSQL_OPTIMIZE_LEVEL attribute, 1-84
PLSQL_TIMEOUT attribute, 1-85
Preallocate attribute, 1-17

PrivateCommands attribute, 1-58
purging log files, 1-36
PWD, 1-64
PWD attribute, 1-64
PWDCrypt, 3-118
PWDCrypt attribute, 1-59

Q

QueryThreshold attribute, 1-60

R

RACCallback attribute, 1-100
RAM
  usage policy, 3-5
RAM policy
  ttAdmin, 3-3
ramPolicy, 3-5
Range optimizer flag, 2-96
read-only isolation mode, 1-52
ReceiverThreads attribute, 1-41
  attributes
    ReceiverThreads, 1-41
RecoveryThreads attribute, 1-42
removing database, 3-45
replication
  ttRepSubscriberStateSet procedure, 2-138
  ttRepSubscriberSync procedure, 2-140
  ttRepSubscriberWait procedure, 2-142
  ttRepSyncGet, 2-144
  ttRepSyncSet, 2-147
Replication Agent
  ttReplicationStatus procedure, 2-120
  ttRepPolicyGet procedure, 2-122
  ttRepPolicySet procedure, 2-124
  ttRepQueryThresholdGet procedure, 2-126
  ttRepQueryThresholdSet procedure, 2-127
  ttRepStart procedure, 2-129
  ttRepStateSet procedure, 2-135
  ttRepStop procedure, 2-137
  ttRepSubscriberStatus procedure, 2-150
  ttRepXactStatus procedure, 2-154
  ttRepXactTokenGet procedure, 2-156
ReplicationApplyOrdering attribute, 1-18
ReplicationParallelism attribute, 1-19
ReplicationTrack attribute, 1-61
RepStateGet, 2-130
restoring databases, 3-102
Rowid optimizer flag, 2-96
row-level locking
  attribute setting, 1-54
  optimizer flag, 2-95, 2-97
running SQL interactively, 3-47

S

Scan optimizer flag, 2-95
serializable isolation mode, 1-52
ServersPerDSN attribute, 1-113
ServerStackSize attribute, 1-114
ShowJoinOrder optimizer flag, 2-96
sizing databases, 2-160, 3-108
SQL, running interactively, 3-47
SQLExecDirect
  example, 2-1
SQLQueryTimeout attribute, 1-62
statistics
  clearing, 2-81
  estimating, 2-83
  getting, 2-85
  modifying explicitly (column), 2-89, 2-92
  modifying explicitly (table), 2-102
  ttOptClearStats procedure, 2-81
  ttOptEstimateStats procedure, 2-83
  ttOptGetColStats procedure, 2-85
  updating explicitly, 2-106
syslog, 3-42

T

table size
  estimating, 2-160, 3-108
tables
  ttBulkCp, 3-14
TBL_STATS system table
  updating explicitly, 2-106
TblLock optimizer flag, 2-96, 2-97
TCP_Port, 1-102, 1-103
TCP_Port attribute, 1-102
TCP_Port2 attribute, 1-103
Temporary attribute, 1-20
temporary database, 1-20
TempSize attribute, 1-43
TempWarnThreshold attribute, 1-63
TimesTen Version
  displaying, 3-119
TmpHash optimizer flag, 2-96
TmpRange optimizer flag, 2-96
TmpTable optimizer flag, 2-96
trace information display, 3-114
tracing
  disabling, 3-115
  enabling, 3-115
  options, 3-115
transaction commit
  and DurableCommits attribute, 1-51
transaction log, 3-128
transaction log API
  ttSetUserColumnID procedure, 2-158
  ttSetUserTableID procedure, 2-159
ttAdmin, 3-3
ttAgingLRUConfig, 2-2
ttAgingScheduleNow, 2-5
ttApplicationContext, 2-7
ttBackup, 3-11
ttBookmark, 2-11
ttBulkCp, 3-14
TTC_FailoverPortRange, 1-104
TTC_FailoverPortRange attribute, 1-104
TTC_Server, 1-105

TTC_Server attribute, 1-105
TTC_Server_DSN, 1-107
TTC_Server_DSN attribute, 1-107
TTC_Server_DSN2, 1-108
TTC_Server_DSN2 attribute, 1-108
TTC_Server2, 1-106
TTC_Server2 attribute, 1-106
TTC_Timeout, 1-109
TTC_Timeout attribute, 1-109
ttCacheAgingIntervalSet, 2-21
ttCacheAutorefreshStatsGet, 2-12, 2-14
ttCacheAwtMonitorConfig, 2-18
ttCacheAwtThresholdGet, 2-20
ttCacheAWTThresholdSet, 2-21
ttCacheConfig, 2-22
ttCacheDbCgStatus, 2-26
ttCachePolicyGet, 2-29
ttCachePolicySet, 2-31
ttCachePropagateFlagSet, 2-33
ttCacheSqlGet, 2-34
ttCacheStart, 2-36
ttCacheStop, 2-37
ttCacheUidGet, 2-38
ttCacheUidPwdSet, 2-38, 2-39
ttCheck, 3-29
ttCkpt, 2-40
ttCkptBlocking, 2-42
ttCkptConfig, 2-44
ttCkptHistory, 2-47
ttCompact, 2-51
ttCompactTS, 2-52
ttConfiguration, 2-53
ttContext, 2-54
ttCWAdmin, 3-32
ttDaemonLog, 3-38
ttDataStoreStatus, 2-55
ttDestroy, 3-45
ttDurableCommit, 2-57
ttGridAttach, 2-58
ttGridCGGlobalResume, 2-66
ttGridcheckOwner, 2-60
ttGridCreate, 2-61
ttGridDestroy, 2-62
ttGridDetach, 2-63
ttGridDetachAll, 2-64
ttGridDetachList, 2-65
ttGridGlobalCGSuspend, 2-67
ttGridInfo, 2-68
ttGridNameSet, 2-70
ttGridNodeStatus, 2-71
ttHostNameGet, 2-73
ttHostNameSet, 2-74
ttIsql, 3-47
ttLockLevel, 2-75
ttLockWait, 2-76
ttLogHolds, 2-78
ttMigrate, 3-72
ttOptClearStats, 2-81
ttOptEstimateStats, 2-83
ttOptGetColStats, 2-85
ttOptGetFlag, 2-86
ttOptGetMaxCmdFreeListCnt, 2-87
ttOptGetOrder, 2-88
ttOptSetFlag, 2-94
ttOptSetMaxCmdFreeListCnt, 2-98
ttOptSetMaxPriCmdFreeListCnt, 2-99
ttOptSetOrder, 2-100
ttOptSetTblStats, 2-102
ttOptShowJoinOrder, 2-104
ttOptUpdateStats, 2-106
ttOptUseIndex, 2-109
ttPLSQLMemoryStats, 2-111
ttRamPolicyGet, 2-113
ttRamPolicySet, 2-115
ttRedundantIndexCheck, 2-117
ttRepDeactivate, 2-119
ttReplicationStatus, 2-120
ttRepPolicyGet, 2-122
ttRepPolicySet, 2-124
ttRepQueryThresholdGet, 2-126
ttRepQueryThresholdSet, 2-127
ttRepStart, 2-129
ttRepStateSet, 2-135
ttRepStateStore, 2-133
ttRepStop, 2-137
ttRepSubscriberStateSet, 2-138
ttRepSubscriberStatus, 2-150
ttRepSubscriberSync, 2-140
ttRepSubscriberWait, 2-142
ttRepSyncGet, 2-144
ttRepSyncSet, 2-147
ttRepSyncSubscriberStatus, 2-150
ttRepTransmitGet, 2-151
ttRepTransmitSet, 2-152
ttRepXactStatus, 2-154
ttRepXactTokenGet, 2-156
ttRestore, 3-102
ttSchema, 3-104
ttSetUserColumnID, 2-158
ttSetUserTableID, 2-159
ttSize, 2-160, 3-108
ttSQLCmdCacheInfo, 2-162
ttSQLCmdCacheInfoGet, 2-164
ttSQLCmdQueryPlan, 2-165
ttStatus, 3-110
ttSyslogCheck, 3-113
ttTail, 3-114
ttTraceMon, 3-115
ttUser, 1-59
ttuser, 3-118
ttVersion, 2-168, 3-119
ttWarnOnLowMemory, 2-169
ttXactIdGet, 2-170
ttXactLog, 3-128
ttXlaBookmarkCreate, 2-171
ttXlaBookmarkDelete, 2-172
ttXlaSubscribe, 2-173
ttXlaUnsubscribe, 2-174
TypeMode attribute, 1-21

U

UID, 1-64
UID attribute, 1-64
updating statistics explicitly, 2-106
user table ID
  definition, 2-159
utilities
  access control, 3-1
  authentication, 3-1
  privileges, 3-1
  ttAdmin, 3-3
  ttBackup, 3-11
  ttBulkCp, 3-14
  ttCheck, 3-29
  ttCWAdmin, 3-32
  ttDestroy, 3-45
  ttIsql, 3-47
  ttMigrate, 3-72
  ttRestore, 3-102
  ttSchema, 3-104
  ttSize, 3-108
  ttStatus, 3-110
  ttSyslogCheck, 3-113
  ttTail, 3-114
  ttTraceMon, 3-115
  ttVersion, 3-119
  ttXactLog, 3-128

W

WaitForConnect attribute, 1-65