Oracle® TimesTen In-Memory Database

Reference

11*g* Release 2 (11.2.2)

E21643-26

March 2015

ORACLE®

Oracle TimesTen In-Memory Database Reference, 11*g* Release 2 (11.2.2)

E21643-26

Contents

2 Built-In Procedures

3 Utilities

Preface

Oracle TimesTen In-Memory Database (TimesTen) is a relational database that is memory-optimized for fast response and throughput. The database resides entirely in memory at runtime and is persisted to disk storage for the ability to recover and restart. Replication features allow high availability. TimesTen supports standard application interfaces SQL, JDBC, ODBC, and ODP.NET, in addition to Oracle interfaces PL/SQL, OCI, and Pro*C/C++. TimesTen is available separately or as a cache for Oracle Database.

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/database-technologies/timesten/documentation/index.html

Conventions

TimesTen supports multiple platforms. Unless otherwise indicated, the information in this guide applies to all supported platforms. The term Windows applies to all supported Windows platforms. The term UNIX applies to all supported UNIX platforms and also to Linux. Refer to the "Platforms" section in *Oracle TimesTen In-Memory Database Release Notes* for specific platform versions supported by TimesTen.

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

This document uses the following text conventions:

Convention	Meaning
italic	Italic type indicates terms defined in text, book titles, or emphasis.

Convention	Meaning
`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.
`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`*	The first three parts in a release number with or without dots. The first three parts of a release number represent a major TimesTen release. For example, `1122` or `11.2.2` represents TimesTen 11*g* Release 2 (11.2.2).
`DSN`	TimesTen data source name (for the TimesTen database).

Documentation Accessibility

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

Access to Oracle Support

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

What's New

This section summarizes the new features of Oracle TimesTen In-Memory Database 11*g* Release 2 (11.2.2) that are documented in this guide and provides links to more information.

New features in Release 11.2.2.7.0

- The `DDLReplicationLevel` and `DDLReplicationAction` connection attributes control what objects that are created or dropped by DDL statements are automatically replicated to the databases involved in an active standby pair replication scheme. For this release, a new level of 3 (not the default) for the `DDLReplicationLevel` connection attribute adds replication of the `CREATE VIEW` or `DROP VIEW` statements, the `CREATE SEQUENCE` or `DROP SEQUENCE` statements, and the result of running the `UidPwdSet` built-in procedure to set the Oracle password for the cache manager for an active standby pair. See "DDLReplicationAction" on page 1-55, "DDLReplicationLevel" on page 1-56, "ttCacheUidPwdSet" on page 2-53, and "ttAdmin" on page 3-3 for details.
- There is a new user error log message designation `RECOVERY`, for messages that report on TimesTen automatic recovery status. This is not a category that you can control through the `ttDaemonLog` utility; the messages cannot be disabled.
- When using Oracle Clusterware, you must execute the new `ttCWAdmin -reauthenticate` command after modifying any user name or password to enable Oracle Clusterware to store these new user names and passwords. For full details, see "Changing user names or passwords when using Oracle Clusterware" in the *Oracle TimesTen In-Memory Database Replication Guide*. Also, see "ttCWAdmin" on page 3-42.
- You can use the `ttIsql edit` command to edit a file or edit `ttIsql` commands in a text editor. The `ttIsql edit` command starts a text editor such as `emacs`, `gedit`, or `vi`. For more information, see "Using the ttIsql edit command" in the *Oracle TimesTen In-Memory Database Operations Guide*. Also, see "ttIsql" on page 3-56.
- The `ttIsql` utility now also includes the command `waitforresult`. The command is similar to the `waitfor` command, except that the result can have 1 or more columns. See "ttIsql" on page 3-56.

New features in Release 11.2.2.6.0

- You can manage the size of the reclaim buffer for the cache agent used to process autorefresh. You can also manage the size of the reclaim buffer for the replication agent when using an active standby pair replication scheme that includes autorefresh cache groups. See "ttDBConfig" on page 2-77.

- Tools have been added to enable control over read optimization during periods of concurrent write operations. For more details, see "ttDbWriteConcurrencyModeGet" on page 2-80 and "ttDbWriteConcurrencyModeSet" on page 2-82.

New features in Release 11.2.2.5.0

- The `ttStats` utility is added to TimesTen to support collection and comparison of snapshots of database metrics.
- You can now set the value for the `cacheAWTMethod` first connection attribute and with the `ttDBConfig` built-in procedure. You can set the `CacheParAwtBatchSize` parameter to configure a threshold value for the number of rows included in a single batch. See "ttDBConfig" on page 2-77.
- At certain times, you may execute large transactions, such as for the end of the month, the end of a quarter, or the end of the year transactions. You may also have situations where you modify or add a large amount of data in the Oracle database over a short period. For read-only, autorefresh cache groups, TimesTen could potentially run out of permanent space when an autorefresh operation applies either one of these cases. Therefore, for these situations, you can configure an autorefresh transaction limit, where the large amount of data is broken up, applied, and committed over several smaller transactions. See "ttCacheAutorefIntervalStatsGet" on page 2-13 and "ttCacheAutorefreshXactLimit" on page 2-25.
- Prolonged use or a heavy workload of the change log tables can result in fragmentation of the tablespace. To prevent degradation of the tablespace from fragmentation of the change log tables, TimesTen calculates the fragmentation for the tablespace and provides a way to defragment the tablespace and reclaim the space. For more details, see "ttCacheAutorefreshLogDefrag" on page 2-18.
- In this release you can specify the amount of memory to allocate in the transaction commit buffer for reclaim records. For details, see the connection attribute "CommitBufferSizeMax" on page 1-50 and the built-in procedures "ttCommitBufferStats" on page 2-64 and "ttCommitBufferStatsReset" on page 2-66.
- In this release, you can use the `AutorefreshLogDefrag` built-in procedure to compact the trigger log space for a cache autorefresh table. See "ttCacheAutorefreshLogDefrag" on page 2-18.

New features in Release 11.2.2.4.0

- In a `ttRepAdmin -duplicate` operation, now you can specify a local or remote IP address for the destination of the duplicate, using the options `-localIP` and `-remoteIP`. See "Duplicate a database" on page 3-108.
- You can use the new Index Advisor to receive recommendations about indexes that could improve the performance of a specific SQL workload. The Index Advisor consists of several built-in procedures. The procedures that comprise the Index Advisor are:
  - `ttIndexAdviceCaptureDrop`
  - `ttIndexAdviceCaptureEnd`
  - `ttIndexAdviceCaptureInfoGet`
  - `ttIndexAdviceCaptureOutput`

  - ttIndexAdviceCaptureStart

- A new tool, the TimesTen Cache Advisor, enables Oracle Database customers to determine whether the performance of an existing Oracle Database application can be improved if the application is used with TimesTen Cache. Cache Advisor generates recommendations of TimesTen cache group definitions based on the SQL usage in the Oracle Database application. For more information, see ttCacheAdvisor.
- New tools have been added that enable you to load the results of a SQL query from a back-end Oracle database into a single table on TimesTen without creating a cache grid, cache group, and cache table to contain the results. TimesTen provides the tools that execute a user provided SELECT statement on an Oracle database and load the result set into a table on TimesTen. The ttIsql utility is enhanced to include the LoadFromOracle command. In addition, two new built-in procedures are in this release as part of this feature:
  - ttLoadFromOracle
  - ttTableSchemaFromOraQueryGet

New features in Release 11.2.2.2.0

- The ttIsql utility now supports autovariables, showcurrenttime as a set/show attribute, the IF command, and WHENEVER error handling. See more details in the description of the ttIsql utility.
- The new Automatic parallel replication allows for the parallel replication and application of transactions changes to the receiving nodes in a replication scheme. See Chapter 1, "Connection Attributes" for more details on parallel replication.

New features in Release 11.2.2.1.0

- The ability to determine the current space usage of a table using the ttComputeTabSizes built-in procedure.
- The new Automatic parallel replication allows for the parallel replication and application of transactions changes to the receiving nodes in a replication scheme. See Chapter 1, "Connection Attributes" for more details on parallel replication.
- You can configure parallel propagation of changes in AWT cache tables to the corresponding Oracle database tables. See Chapter 1, "Connection Attributes" for more details on parallel propagation.
- The maximum value of the LogBufMB and LogFileSize connection attributes has increased to 64 GB on 64-bit systems.

1

Connection Attributes

The ODBC standard defines four connection attributes:

- DSN
- Driver
- UID
- PWD

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 connection 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.

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

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, this chapter describes each attribute in detail.

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

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
`Driver`	Specifies the TimesTen ODBC Driver Manager.	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)
`RangeIndexType`	Determines whether user-created range indexes are T-tree indexes or B-tree indexes.	`1` (Range indexes are T-tree indexes.)
`ReplicationApplyOrdering`	Enables automatic or user-defined track-based parallel replication.	`0` (Starts automatic parallel replication)
`ReplicationParallelism`	Specifies the number of tracks available for user-defined parallel replication.	`1`

Table 1–1 (Cont.) Data store attributes

Name	Description	Default
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)
CkptReadThreads	Controls the number of threads used to read a checkpoint file when loading the database into memory.	1
Connections	Indicates the upper bound on the number of user-specified concurrent connections to the database.	The lesser of 2000 or the number of semaphores specified in the SEMMSL kernel parameter
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.	4
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)
LogPurge	Specifies that unneeded transaction log files are deleted during a checkpoint operation.	1 (True)
MemoryLock	enables 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 as determined from the PermSize value

Table 1–3 General connection attributes

Name	Description	Default
CommitBufferSizeMax	Specifies the maximum size of the commit buffer in the transaction control block.	16 KB
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)
DDLReplicationAction	Determines whether a table or sequence is included in an active standby pair replication scheme when it is created, which can only occur if the DDLReplicationLevel connection attribute is set to 2 or 3.	INCLUDE

Table 1–3 (Cont.) General connection attributes

Name	Description	Default
DDLReplicationLevel	Enables replication of data definition language (DDL) statements in an active standby replication scheme.	2 (Replication of certain objects enabled)
Diagnostics	Specifies whether diagnostic messages are generated.	1 (Messages are generated)
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).	0 (Row-level locking)
LockWait	Enables an application to configure the lock wait interval for the connection.	10 seconds
MatchLogOpts	Specifies that value used for the LogPurge attribute 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 1-73.	Specify the password that corresponds with the specified UID. When caching data from an Oracle database, 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 a warning and throws an SNMP trap if a query times out before executing.	0 (No warning is returned)
ReplicationTrack	Assigns a connection to a replication track.	None

Table 1–3 (Cont.) General connection attributes

Name	Description	Default
SQLQueryTimeout	Specifies the time limit in seconds within which the database should execute SQL statements.	0 (No timeout)
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 (percent)
UID See "UID and PWD" on page 1-73.	Specify a user name that is defined on the TimesTen server. When caching data from an Oracle database, 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	Enables or disables whether PL/SQL.	1 (Enables PL/SQL)
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 the PL/SQL compiler uses 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 TimesTen Cache first connection attributes

Name	Description	Default
CacheAWTMethod	Enables the AWT propagation method to be used on Oracle database tables.	1 (PL/SQL)

Table 1–8 TimesTen Cache database attributes

Name	Description	Default
CacheAWTParallelism	Indicates the number of threads necessary to apply changes to the Oracle database.	1
CacheGridEnable	Enables cache grid.	1 (Enabled)
CacheGridMsgWait	Sets the maximum message wait time in seconds.	60

Table 1–9 TimesTen Cache general connection attributes

Name	Description	Default
DynamicLoadEnable	Enables or disables transparent load of data from an Oracle database to dynamic cache groups.	1 (Enables Dynamic cache group load)
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 database 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

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

Name	Description	Default
PassThrough	Specifies which SQL statements are executed locally in TimesTen and which SQL statements are passed through to the Oracle database for execution.	0
RACCallback	Specifies whether to enable or disable the installation of Application Failover (TAF) and Fast Application Notification (FAN) callbacks.	1 (Install callbacks)

Table 1–10 TimesTen Client connection attributes

Name	Description	Default
TCP_Port	The port number on which the TimesTen server is listening.	None
TCP_Port2	The port number on which the TimesTen 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 TimesTen 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 TimesTen 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.	60

Table 1–11 TimesTen Server connection attributes

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

Data store attributes

Data store attributes are set at data store creation time. The data store attributes are listed in Table 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 (DSN) 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. The ODBC driver manager and TimesTen use this information.

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`.

> **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 1-76.

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 supported character set names, see "Supported character sets" below.

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

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.

Driver

The `Driver` attribute specifies the name of the TimesTen ODBC Driver.

For example, on Windows systems the value can be `TimesTen Data Manager 11.2.2` or `TimesTen Client 11.2.2`.

On UNIX systems, the value of the `Driver` attribute is the path name of the TimesTen ODBC Driver shared library file. The file resides in the in`stall_dir/lib` directory.

Required privilege

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

Setting

Set `Driver` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `odbc.ini` file	`Driver`	Specifies the path name for the TimesTen ODBC Driver shared library file.
Windows ODBC Data Source Administrator	Select a driver from the **Create New Data Source** dialog.	Specifies the Client or Data Manager driver for TimesTen and the release.

LogDir

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

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 (default) - Does not preallocate file system space for database when creating the database. 1 - Preallocates file system space for the database.
Windows ODBC Data Source Administrator	**Preallocate** check box	**unchecked** (default) - Does not preallocate file system space for database when creating the database. **checked** - Preallocates file system space for the database.

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

RangeIndexType

The `RangeIndexType` attribute specifies whether user-created range indexes use T-tree or B-tree indexes.

> **Note:** Once a database is created with the `RangeIndexType` set to `0`, you cannot use the database with a pre-11.2.2.5.0 TimesTen release.

For more information on indexes, see "Overview of index types" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

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

Setting

Set `RangeIndexType` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `odbc.ini` file	`RangeIndexType`	`0` - All user-created range indexes are B-tree indexes. 1 (default)- All user-created range indexes use t-tree indexes.
Windows ODBC Data Source Administrator	**Range Index Type** field	`B-tree` - All user-created range indexes are B-tree indexes. `T-tree` (default) - All user-created range indexes use t-tree indexes.

ReplicationApplyOrdering

Enables parallel replication when used with the `ReplicationParallelism` attribute. With parallel replication, multiple transmitters on the master send to multiple receivers on the subscriber.

- Automatic parallel replication: Parallel replication over multiple threads that automatically enforces transactional dependencies and all changes applied in commit order. This is the default.
- Automatic parallel replication with disabled commit dependencies: Parallel replication over multiple threads that automatically enforces transactional dependencies, but does not enforce transactions committed in the same order on the subscriber database as on the active database. You can also increase replication throughput by applying transactions to specific tracks.
- User-defined parallel replication: For applications that use classic replication schemes, 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 increase replication throughput by specifying the number of transaction tracks and apply specific transactions to each track. All tracks are read, transmitted and applied in parallel.

For more details on configuring parallel replication, see "Configuring parallel replication" in the *Oracle TimesTen In-Memory Database Replication Guide*.

This attribute also sets parallel propagation for AWT cache groups. By default, this attribute enables parallel propagation of updates to the Oracle database. To learn more about parallel AWT caching, see "Configuring parallel propagation to Oracle Database tables" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

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 automatic parallel replication. Automatic parallel replication is available for both classic and active standby pair replication schemes. (default) `1` - Specifies user-defined track-based parallel replication. User-defined parallel replication is available only for classic replication schemes. `2` - Specifies automatic parallel replication with disabled commit dependencies.

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	**Replication Apply Ordering** pulldown list	`0` - Specifies automatic parallel replication. Automatic parallel replication is available for both classic and active standby pair replication schemes. (default) `1` - Specifies user-defined track-based parallel replication. User-defined parallel replication is available only for classic replication schemes. `2` - Specifies automatic parallel replication with disabled commit dependencies.

Restrictions

Restrictions when using automatic parallel replication with disabled commit dependencies:

- The replication scheme must be an active standby pair that uses asynchronous replication. Classic replication schemes are not supported.
- The replication scheme cannot contain cache groups.
- This is only supported for TimesTen Release 11.2.2.8 and following for both the active and standby masters. Both the active and standby masters must have commit dependencies disabled.
- XLA is not supported.

All data stores in the replication scheme must use the same setting.

Active standby pairs cannot use user-defined parallel replication.

ReplicationParallelism

This attribute specifies the number of tracks, or the number of transmitter/receiver pairs, used for parallel replication.

The default value for this attribute is 1. This value indicates that single-threaded replication occurs. If the value is greater than 1, the LogBufParallelism first connection attribute must be an integral multiple of ReplicationParallelism.

To configure parallel replication, set this attribute to a value from 2 to 32, indicating the number of transmitter/receiver pairs.

If the CacheAWTParallelism attribute is set to 1 or not set, the maximum allowable value for ReplicationParallelism is 16.

To learn more about parallel replication, see "Configuring parallel replication" in the *Oracle TimesTen In-Memory Database 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 32, indicating the number of tracks to replicate in parallel. The default is 1, single-threaded replication.
Windows ODBC Data Source Administrator	**Replication Parallelism** field	*n* - A value between 1 and 32, indicating the number of tracks to replicate in parallel. The default is 1, single-threaded replication.

Restrictions

Restrictions and things to consider when specifying parallel replication include:

- When parallel replication is enabled, the Description column of the ttLogHolds built-in procedure displays one row per track per subscriber node.
- We recommend setting the value of this attribute to a value no greater than half the value of the LogBufParallelism attribute. If you specify more replication tracks than log buffer threads, some replication tracks can remain empty.
- Synchronous replication, including TWOSAFE and RETURN RECEIPT replication, is not supported with user-defined parallel replication.
- Active standby pairs are not supported with user-defined parallel replication.
- Parallel replication with disabled commit dependencies is not supported with user-defined parallel 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` (default) - Creates a permanent database. `1` - Creates a temporary database.
Windows ODBC Data Source Administrator	**Temporary** check box	**unchecked** (default) - Creates a permanent database. **checked** - Creates temporary a 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. When creating a DSN, use the default setting, Oracle type mode.

When caching data from an Oracle database 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` (default) - Oracle type mode `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` (default) - Oracle type mode `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 and then connect with `ADMIN` privileges. (No privileges are required to change `AutoCreate` and `ForceConnect`.)

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.

Also see "Overwrite" on page 1-44.

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` (default) - Creates a new database if the specified database does not exist.
Windows ODBC Data Source Administrator	**AutoCreate** check box	**unchecked** - Does not create new database if database does not exist. **checked** (default) - Creates a new database if database does not exist.

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.

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. The default is `600`. If you do not specify this attribute, TimesTen uses the default value (`600`) for database creation. For an existing database, TimesTen will use the stored value. If the attribute is specified, but you do not supply a value, the value of `0` is used. Specifying a value of `-1` is equivalent to omitting this attribute. If you specify a value of `-1`, the default value (`600`) is used for database creation, otherwise the stored value is used.

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. The default is `600`. If you do not specify this attribute, TimesTen uses the default value (`600`) for database creation. For an existing database, TimesTen will use the stored value. If the attribute is specified, but you do not supply a value, the value of `0` is used.

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.

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`. If you do not specify this attribute, TimesTen uses the default value (0) for database creation. For an existing database, TimesTen will use the stored value. If the attribute is specified, but you do not supply a value, TimesTen uses the default value (0). Specifying a value of `-1` is equivalent to omitting this attribute. If you specify a value of `-1`, TimesTen uses the default value (`0`) is used for database creation. If the database already exists, TimesTen uses the stored value.

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	**Ckpt LogVolume** field	Specify the amount of data in MBs that can accumulate in the transaction log file between background checkpoints. The default is `0`. If you do not specify this attribute, TimesTen uses the default value (`0`) for database creation. For an existing database, TimesTen will use the stored value. If the attribute is specified, but you do not supply a value, TimesTen uses the default value (0).

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.

All background checkpoints and by checkpoints initiated by the `ttCkpt` and `ttCkptBlocking` built-in procedures use the rate specified by this connection attribute. *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), TimesTen uses the stored setting. 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. If you do not specify this attribute, TimesTen uses the default value (`0`) for database creation. TimesTen uses the stored value for existing databases. If the attribute is specified, but you do not supply a value, the value of `0` is used. Specifying a value of `-1` is equivalent to omitting this attribute. If you specify a value of `-1`, the default value (`0`) is used for database creation, otherwise the stored value is used.
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. If you do not specify this attribute, TimesTen uses the default value (`0`) for database creation. TimesTen uses the stored value for existing databases. If the attribute is specified, but you do not supply a value, TimesTen uses the default value (`0`).

CkptReadThreads

Determines the number of threads used to read the checkpoint file when loading the database into memory, such as in first connection or recovery operations.

When the `CkptReadThreads` attribute is set to `1`, TimesTen reads checkpoint files serially. When the `CkptReadThreads` attribute is set to a value greater than `1`, TimesTen uses the specified number of threads to read checkpoint files concurrently. When the `CkptReadThreads` attribute is set to `0` or unspecified, the previously specified value is used.

Required privilege

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

Setting

Set `CkptReadThreads` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX `odbc.ini` file	`CkptReadThreads`	*n* - The number of threads to use when reading the checkpoint files during the loading of the database into memory. Takes an integer value of `0` or greater (maximum = $2^{31}-1$). Default is `1`.
Windows ODBC Data Source Administrator	**Checkpoint Read Threads** field	*n* - The number of threads to use when reading the checkpoint files during the loading of the database into memory. Takes an integer value of `0` or greater (maximum = $2^{31}-1$). Default is `1`.

Notes

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

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

Connections

Indicates the upper bound on the number of user-specified concurrent connections to the database. TimesTen allocates one semaphore for each expected connection. If the number of connections exceeds the value of this attribute, TimesTen returns an error.

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 you should use the default number of semaphores. The maximum number of allowed connections is 2047. TimesTen reserves 47 connections for internal use, leaving a maximum of 2000 connections available to applications. As a guideline, set this value to the maximum number of expected application connections plus ten percent.

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.

When you enable PL/SQL (`PLSQL=1`), there is both a fixed and per connection overhead allocated from the PL/SQL segment, even if you do not use PL/SQL. For details, see "PLSQL_MEMORY_SIZE" on page 1-88.

> **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

Only the instance administrator can 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`	`0` or no value - Indicates that the default value is used. The default value is the lesser of `2000` or the number of semaphores specified in the `SEMMSL` kernel parameter. An integer from `1` through `2000` -The value represents the maximum number of connections.
Windows ODBC Data Source Administrator	**Connections** field	`0` or no value - Indicates that the default value is used. The default value is `2000`. An integer from `1` through `2000` -The value represents the 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.

The `ttConfiguration` built-in procedure does not return the value of the `ForceConnect` attribute.

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` (default) - Do not allow connection to failed database if it is not properly restored from the corresponding subscriber database. `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** (default) - Do not allow connection to failed database if it is not properly restored from the corresponding subscriber database. **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` (default) - 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.
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** (default) - 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.

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.

If you increase the value of LogBufMB, ensure the value of LogFileSize is greater than or equal to the value of LogBufMB (LogFileSize >= LogBufMB).

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, TimesTen uses the value stored in the database. If not set and the database is being created, TimesTen uses the default value (64). On 32-bit systems, the maximum log file is 1024 MB (1 GB). On 64-bit systems, the maximum value is 65,536 MB (64 GB).
Windows ODBC Data Source Administrator	**Log Buffer Size (MB)** field	Size of log buffer, in megabytes. If not set and the database exists, TimesTen uses the value stored in the database. If not set and the database is being created, TimesTen uses the default value (64). On 32-bit systems, the maximum log file is 1024 MB (1 GB). On 64-bit systems, the maximum value is 65,536 MB (64 GB).

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	**Log Buffer Parallelism** 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 1 MB. 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.

If you specify a value of zero, TimesTen uses the default transaction log file size if the database does not exist. If the database exists, TimesTen uses the current specified transaction log file size.

Set the value of `LogFileSize` to be larger than or equal to the value of `LogBufMB` (`LogFileSize >= LogBufMB`).

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. On 32-bit systems, the maximum log file is 1024 MB (1 GB). On 64-bit systems, the maximum value is 65,536 MB (64 GB).
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. On 32-bit systems, the maximum log file is 1024 MB (1 GB). On 64-bit systems, the maximum value is 65,536 MB (64 GB).

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` (default) - 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). `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` (default) - 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). `2` - Write data to transaction log files using synchronous writes such that explicit sync operations are not needed.

See also

`DurableCommits`

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, TimesTen writes the contents of the transaction log files 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.

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 (default) - Removes old transaction log files at connect and checkpoint.
Windows ODBC Data Source Administrator	**LogPurge** check box	**unchecked** - Does not remove old transaction log files at connect and checkpoint. **checked** (default) - Removes old transaction log files at connect and checkpoint.

See also

MatchLogOpts

MemoryLock

On Solaris, Linux and Windows 64-bit systems, TimesTen enables 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 subsystem 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 have the `proc_lock_memory` privilege to set `MemoryLock` to `1` or `2`. No special privileges are required to set `MemoryLock` to `3` or `4`. Setting `MemoryLock` to `3` or `4` enables use of Solaris "intimate shared memory (ISM)".

To view privileges, use:

```
% ppriv $$
```

To add the privilege for user ID `timesten`, a root user uses:

```
# usermod -K defaultpriv=basic,proc_lock_memory timesten
```

After adding the privilege, the `timesten` user must log in to a new shell, unload all TimesTen databases from memory and restart the TimesTen daemons.

To restart the TimesTen daemons, in the new login shell, use:

```
% ttDaemonadmin -restart
```

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` (default) - Does not lock memory. `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` (default) - Does not lock memory. `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 if 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 1-26). TimesTen ignores the Overwrite attribute 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 (default) - TimesTen does not overwrite an existing database with the same path name. 1 - TimesTen overwrites an 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 to use the default size. 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*.

Also see information about the `TempSize` connection attribute.

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 have two cores (or hardware threads) or more 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 TimesTen 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 64 MB, 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 the temporary partition of the database, in MB. Minimum size is 32 MB on all platforms.
Windows ODBC Data Source Administrator	**Temporary Data Size** field	*n* - Size of the temporary partition of the database, in MB. 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.

CommitBufferSizeMax

CommitBufferSizeMax indicates the total amount of memory in MB allocated to the transaction commit buffer. Set this attribute to handle the size of reclaim records.

You can use the ALTER SESSION SQL statement, described in *Oracle TimesTen In-Memory Database SQL Reference*, to assign or change the maximum size of the commit buffer within a session. The new value takes effect when a new transaction starts.

```
ALTER SESSION SET COMMIT_BUFFER_SIZE_MAX = n;
```

You can see the configured maximum for the commit buffer by calling the ttConfiguration built-in procedure.

For more information on reclaim operations, including details about setting the commit buffer size, see "Transaction reclaim operations" in the *Oracle TimesTen In-Memory Database Operations Guide*. Also see information about the ttCommitBufferStats and the ttCommitBufferStatsReset built-in procedures.

Required privilege

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

Setting

Set CommitBufferSize as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX odbc.ini file	CommitBufferSizse	0 - Commit buffer gets configured with a default size (128K). *n* - Integer value. Minimum user configured size is 1 (MB), This value should be configured to a value much smaller than TempSize.
Windows ODBC Data Source Administrator	**Maximum Commit Buffer Size** field	*n* - Integer value. Minimum user configured size is 1 (MB), This value should be configured to a value much smaller than TempSize.

Notes

When you call the built-in procedure ttCommitBufferStats, the commit buffer statistics are expressed in bytes. However, the ttConfiguration built-in procedure output and the value set by the connection attribute CommitBufferSizeMax are expressed in MB.

ConnectionName

This attribute is also available as a Client connection attribute.

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

TimesTen uses the symbolic name to help identify the connection in various 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 `conn`, 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 `conn`, 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 data definition language (DDL) 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, user procedures, and indexes)
- `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` (default) - 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. `1` - Traditional TimesTen style behavior. Execution of DDL statements does not trigger implicit transaction commits.
Windows ODBC Data Source Administrator	**DDLCommitBehavior** field	`0` (default) - 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. `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 `INSERT` statements 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
```

DDLReplicationAction

Determines whether a table or a sequence is included in an active standby pair replication scheme when created. The table can be included if the DDLReplicationLevel connection attribute is set to 2 or 3. The sequence can be included if the DDLReplicationLevel connection attribute is set to 3.

Replication of DDL operations is enabled (with restrictions) by the set value of the DDLReplicationLevel connection attribute. For more details, see "DDLReplicationLevel" on page 1-56.

The value may be modified by an ALTER SESSION SQL statement, described in *Oracle TimesTen In-Memory Database SQL Reference*. For example:

```
ALTER SESSION SET DDL_REPLICATION_ACTION='EXCLUDE';
```

Values set by ALTER SESSION override the value set by this attribute.

For examples of altering an active standby pair, see "Altering an Active Standby Pair" in the *Oracle TimesTen In-Memory Database Replication Guide*.

When DDLCommitBehavior=0 (the default), DDL operations are automatically committed. When RETURN TWOSAFE has been specified, errors and timeouts may occur as described in "RETURN TWOSAFE" in the *Oracle TimesTen In-Memory Database Replication Guide*. If a RETURN TWOSAFE timeout occurs, the DDL transaction is committed locally regardless of the LOCAL COMMIT ACTION that has been specified.

To learn more about replicating DDL, see "Changing user names or passwords used by replication" in the *Oracle TimesTen In-Memory Database Replication Guide*.

Required privilege

ADMIN privilege is required if the value of this attribute is INCLUDE.

Setting

Set DDLReplicationAction as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX odbc.ini file	DDLReplicationAction	INCLUDE (default) - When a table or sequence is created, it is automatically added to the active standby pair scheme when the appropriate DDLReplicationLevel value is configured. EXCLUDE - When a table or sequence is created, it is not automatically included in the active standby pair.
Windows ODBC Data Source Administrator	**DDLReplicationAction** field	INCLUDE (default) - When a table or sequence is created, it is automatically added to the active standby pair scheme when the appropriate DDLReplicationLevel value is configured. EXCLUDE - When a table or sequence is created, it is not automatically included in the active standby pair.

DDLReplicationLevel

Enables replication of a subset of data definition language (DDL) statements (with restrictions) in an active standby replication scheme.

When the value of the `DDLReplicationLevel` connection attribute is set to 1, `CREATE` or `DROP` statements for tables, indexes, or synonyms are not replicated to the standby database. However, you can add or drop columns with the `ALTER TABLE ADD` or `DROP COLUMN` to or from a replicated table, and those actions are replicated to the standby database.

When the value of the `DDLReplicationLevel` connection attribute is set to 2 (the default), the following DDL statements (described in *Oracle TimesTen In-Memory Database SQL Reference*) are replicated to the standby and any subscribers:

- `CREATE` or `DROP INDEX`
- `CREATE` or `DROP SYNONYM`
- `CREATE` or `DROP TABLE` (including global temporary tables but not `CREATE TABLE AS SELECT`)

When the value of the `DDLReplicationLevel` connection attribute is set to 3, the following DDL statements (described in *Oracle TimesTen In-Memory Database SQL Reference*) and those replicated when the value is set to 2 are replicated to the standby and any subscribers:

- `CREATE` or `DROP VIEW`
- `CREATE` or `DROP SEQUENCE`
- Replication of the results to the standby master when setting the cache administration user name and password with the `UidPwdSet` built-in procedure. You do not need to stop and restart the cache agent or replication agent to execute the `UidPwdSet` built-in procedure. For more information, see "Changing cache user names or passwords" in the *Oracle TimesTen Application-Tier Database Cache User's Guide* or "ttCacheUidPwdSet" on page 2-53.

The value of this attribute may be modified by an `ALTER SESSION` statement, described in *Oracle TimesTen In-Memory Database SQL Reference*. For example:

```
ALTER SESSION SET DDL_REPLICATION_LEVEL=3;
```

Values set by `ALTER SESSION` override the value set by this attribute.

For examples of altering an active standby pair, see "Altering an Active Standby Pair" in the *Oracle TimesTen In-Memory Database Replication Guide*.

To learn more about replicating DDL, see "Changing user names or passwords used by replication" in the *Oracle TimesTen In-Memory Database Replication Guide*.

Required privilege

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

Setting

Set `DDLReplicationLevel` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX odbc.ini file	DDLReplicationLevel	1 - Replicates ALTER TABLE ADD or DROP COLUMN to the standby database. Does not replicate CREATE and DROP operations for tables, indexes, or synonyms to the standby database. 2 (default) - Replicates creating and dropping of tables, indexes and synonyms. 3 - Replicates creating and dropping of views and sequences and replicates the results of the UidPwdSet built-in procedure.
Windows ODBC Data Source Administrator	**DDL Replication Level** field	1 - Replicates ALTER TABLE ADD or DROP COLUMN to the standby database. Does not replicate CREATE and DROP operations for tables, indexes or synonyms to the standby database. 2 (default) - Replicates creating and dropping of tables, indexes and synonyms. 3 - Replicates creating and dropping of views and sequences and replicates the results of the UidPwdSet built-in procedure.

Restrictions

Replication of DDL operations has these restrictions:

- CREATE TABLE AS SELECT statements are not replicated.
- The CREATE INDEX statement is replicated only when the index is created on an empty table.
- To control whether a table or sequence is included in an active standby pair replication scheme at the time of creation, use the DDLReplicationAction connection attribute.
- Sequences with the CYCLE attribute cannot be replicated.
- Objects are replicated only when the receiving database is of a TimesTen release that supports that level of replication, and is configured for an active standby pair replication scheme. For example, replication of sequences (requiring DDL_REPLICATION_LEVEL=3) to a database release prior to 11.2.2.7.0 is not supported. When DDLReplicationLevel value is set to 3, both the active and standby master databases need to be TimesTen Release 11.2.2.7 or later. When DDL_REPLICATION_LEVEL=2, the receiving database must be at least release 11.2.1.8.0 for replication of objects to be supported.
- All restrictions for the UidPwdSet built-in procedure apply.
- When DDLReplicationLevel=1 or 2, you cannot alter a table to add a NOT NULL column to a table that is part of a replication scheme with the ALTER TABLE ... ADD COLUMN NOT NULL DEFAULT statement. You must remove the table from the replication scheme first before you can add a NOT NULL column to it. However, if DDLReplicationLevel=3, then you can alter a table to add a NOT NULL column to a table that is part of a replication scheme.

Diagnostics

Enables 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` - TimesTen does not generate diagnostic messages. `1` (default) - TimesTen generates base-level diagnostics messages.
Windows ODBC Data Source Administrator	**Diagnostics** field	`0` - TimesTen does not generate diagnostic messages. `1` (default) - TimesTen generates base-level diagnostics messages.

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 Database assigns a number to each parameter occurrence without regard to name duplication. An Oracle Database 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` (default) - SQL statements use the Oracle parameter binding model. `1` - SQL statements use the traditional TimesTen parameter binding model.
Windows ODBC Data Source Administrator	**Duplicate Bind Mode** check box	**unchecked** (default) - SQL statements use the Oracle parameter binding model. **checked** - SQL statements 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. With this setting, TimesTen writes a log record 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, TimesTen writes a log record 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 (default) - TimesTen does not write the transaction log to disk on transaction commit. 1 - TimesTen writes log to disk on transaction commit.
Windows ODBC Data Source Administrator	**Durable Commits** check box	**unchecked** (default) - TimesTen does not write the transaction log to disk on transaction commit. **checked** - TimesTen writes 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, described in *Oracle TimesTen In-Memory Database SQL Reference*. For example:

```
ALTER SESSION SET ISOLATION_LEVEL=serializable;
```

`CREATE CACHE GROUP`, `ALTER CACHE GROUP` and `DROP CACHE GROUP` statements are not supported in serializable isolation mode.

If the passthrough or the propagate TimesTen 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, see "PassThrough" on page 1-107.

With `PassThrough` set to 3, you must use an `ALTER SESSION` statement to permanently modify the isolation level on the Oracle database connection. For example on a connection to the DSN `repdb1_1122`:

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

   ```
   % ttIsql;
   Command> connect "dsn=repdb1_1122;passthrough=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` (default) - Connects to database in read committed mode.
Windows ODBC Data Source Administrator	**Isolation** dropdown list	`0` - Connects to database in serializable isolation mode. `1` (default) - Connects to database in read committed isolation mode.

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 database tables, you must set row-level locking. To `CREATE`, `DROP`, or `ALTER` a user, you can only use row-level locking and thus, the lock level 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

`ADMIN` privilege is required if the value of this attribute is 1.

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` (default) - Transactions access the database using row-level locking. `1` - Transactions access the database by acquiring an exclusive lock on the entire database.
Windows ODBC Data Source Administrator	**DS-Level Locking** check box	**unchecked** (default) - Transactions access the database using row-level locking. **checked** - Transactions access the database by acquiring an exclusive lock on the entire database.

LockWait

Enables 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.

The number of connections to a database can impact the time needed to resolve lock contentions. If you anticipate having many connections to the database, increase the lock wait interval.

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* - Indicates the number of seconds to wait for locking conflict resolution before timing out. The default is `10` seconds.
Windows ODBC Data Source Administrator	**LockWait** field	*s* - Indicates the number of seconds to wait for locking conflict resolution before timing out. The default is `10` seconds.

MatchLogOpts

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

> **Note:** If MatchLogOpts is set to True for the first connection, TimesTen returns an error and the connection fails. Use the attribute with MatchLogOpts connection 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 (default) - Value of LogPurge is used. 1 - TimesTen ignores the value of LogPurge. Instead, values match those of current connections.
Windows ODBC Data Source Administrator	**Match Log Opts** check box	**unchecked** (default) - Value of LogPurge is used. **checked** - TimesTen ignores the value of LogPurge. 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. Also see "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 TimesTen should issue out-of-memory warnings. Default is `90`.
Windows ODBC Data Source Administrator	**Low Memory Warning Thresholds for Permanent Data** field	*p* - Percentage at which TimesTen should issue out-of-memory warnings. 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` (default) - Commands are shared with other connections. `1` - Commands are not shared with any other connection.
Windows ODBC Data Source Administrator	**Private Commands** field	`0` (default) - Commands are shared with other connections. `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. 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 enables 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 enables 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, TimesTen uses the value of the PWD attribute. See "UID and PWD" on page 1-73.

TimesTen does not store the value of the PWD attribute anywhere in the TimesTen system.

See "Required user authentication for utilities" in the description of "UID and PWD" for details about the treatment of passwords when using utilities that require specific privileges.

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 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 nonnegative integer. Default is `0` and indicates that TimesTen does not return a warning.
Windows ODBC Data Source Administrator	**QueryThreshold (secs)** field	A nonnegative integer. Default is `0` and indicates that TimesTen does not return a warning.

ReplicationTrack

When managing track-based parallel replication, you can assign a connection to a replication track. All transactions issued by the connection are assigned to this track, unless the track is altered.

To start track-based 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` either to 1 or 2, depending on if you are using automatic or user-defined parallel replication.

The `Track_ID` column of the `TTREP.REPPEERS` system table (described in *Oracle TimesTen In-Memory Database System Tables and Views Reference*) shows the track associated with the connection.

You can use the `ALTER SESSION` SQL statement, described in *Oracle TimesTen In-Memory Database SQL Reference*, to assign or change the value of this attribute within a session. For example:

```
ALTER SESSION SET REPLICATION_TRACK=4;
```

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.

Restrictions

Restrictions and things to consider when specifying user-defined parallel replication include:

- Synchronous replication, including `TWOSAFE` and `RETURN RECEIPT` replication, is not supported with user-defined parallel replication.
- Active standby pairs are not supported with user-defined parallel replication.

SQLQueryTimeout

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

The value of `SQLQueryTimeout` 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 TimesTen Cache operations that are being processed on an Oracle database. 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 2-220. Also see "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 TimesTen 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 database tables, PWD specifies the TimesTen password while OraclePWD specifies the Oracle password.

Required user authentication for utilities

All utilities that require a password prompt for one.

If a UID connection 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 connection attribute is not recommended in the first call to the utility. For example, the following usage is not recommended:

```
% ttIsql -connStr "DSN=mydsn;UID=terry;PWD=secret";
```

Generally, when no UID connection 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 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` (default) - Waits until connection to database is possible.
Windows ODBC Data Source Administrator	**Wait For Connect** check box	**unchecked** - Does not wait if connection to database fails. **checked** (default) - Waits until connection to database is possible.

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, described in *Oracle TimesTen In-Memory Database SQL Reference*, to change NLS parameters to override the values that are assigned to these attributes at connection time.

ConnectionCharacterSet

ConnectionCharacterSet 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 supported character set names, see "Supported character sets" on page 1-12.

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, unless the database uses the TIMESTEN8 character set.
Windows ODBC Data Source Administrator	**Connection CharacterSet** list	The default value for ConnectionCharacterSet is US7ASCII, unless the database uses the TIMESTEN8 character set.

NLS_LENGTH_SEMANTICS

TimesTen uses the NLS_LENGTH_SEMANTICS attribute 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.

Implicit and explicit conversions between CHAR and NCHAR are supported, unless the database uses the TIMESTEN8 character set.

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 (default) - Errors are not reported when there is a data loss during character type conversion. 1 - Errors are reported when there is a data loss during character type conversion.
Windows ODBC Data Source Administrator	**NLS_NCHAR_ CONV_EXCP** check box	unchecked (default) - Error messages are not reported when there is a data loss during character type conversion. checked - 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.

Operations involving character comparisons support linguistic case-sensitive collating sequences. Case-insensitive sorts may affect DISTINCT value interpretation.

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.

In TimesTen installations where PL/SQL support was enabled at installation time, the default is `PLSQL=1`. Or PL/SQL may be enabled when the database is initially created, or at any first connect afterward. Once PL/SQL support is enabled in a database, you cannot disable it later.

Configuring PL/SQL support in a database results in the installation of several PL/SQL packages that are documented in *Oracle TimesTen In-Memory Database PL/SQL Packages Reference*.

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`, 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`).
- When you enable PL/SQL (`PLSQL=1`), there is both a fixed and per connection overhead allocated from the PL/SQL segment, even if you do not use PL/SQL. For details, see "PLSQL_MEMORY_SIZE" on page 1-88.

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` (default) - Enables PL/SQL for the database.

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

PLSQL_MEMORY_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.

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.

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 space.

The platform specific default memory addresses are:

Operating system	Address
Linux on 32-bit x86 processors	`10000000`
Linux on 64-bit x86 processors	`0000007fa0000000`
32-bit AIX	`c0000000`
64-bit AIX	`06ffffff00000000`
Solaris on 64-bit x86 processors	`0000007fa0000000`
Solaris on 64-bit SPARC processors	`ffffff0000000000`
32-bit Windows	`5B8C0000`
64-bit Windows	`000000005b8c0000`
64-bit HP-UX	`0`

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 `PLSQL_MEMORY_ADDRESS`.
- The value of this attribute is stored persistently by TimesTen. The persistent attribute value is 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_SIZE`.

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. All connections share this memory segment.

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. The persistent attribute value is 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 32 MB on Windows 32-bit systems. The minimum size is 2 MB, 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`.
- When you enable PL/SQL (`PLSQL=1`), there is both a fixed and per connection overhead allocated from the PL/SQL segment, even if you do not use PL/SQL. The minimum fixed memory allocated is approximately 1500 KB. Additionally, approximately 40 KB of memory is allocated per connection. Thus, you can compute an estimated minimum memory setting needed when enabling PL/SQL for `PLSQL_MEMORY_SIZE` as 1500 KB plus (*`number_of_connections * 40`*). If the application uses PL/SQL, we recommend that you allocate twice the estimated minimum required memory for this segment. If the application does not use PL/SQL, you can allocate less than twice the estimated minimum required memory.

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 in MB of the shared memory segment in megabytes. The default size is `32`.

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

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 in Table 1–6, " PL/SQL general connection attributes" and described in detail in this section.

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

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, described in *Oracle TimesTen In-Memory Database SQL Reference*. If the attribute is specified in an ALTER SESSION statement in a database where PLSQL=0, an error is returned. For example

```
ALTER SESSION SET PLSCOPE_STTINGS = 'IDENTIFIERS:ALL' ;
```

- 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 TimesTen In-Memory Database PL/SQL Developer's Guide.*

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 sets directives to control conditional compilation of PL/SQL units, which enables 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'
```

v1 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.

c1 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, described in *Oracle TimesTen In-Memory Database SQL Reference*, to change this attribute within a session.

```
ALTER SESSION SET PLSQL_CCFLAGS = 'v1:c1,v2:c2,...,vn:cn';
```

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, described in *Oracle TimesTen In-Memory Database SQL Reference*. For example:

```
ALTER SESSION SET PLSQL_CONN_MEM_LIMIT = 100;
```

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 `100`.
Windows ODBC Data Source Administrator	**PL/SQL Connection Memory Limit** field	An integer value in MB. Default value is `100`.

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, described in *Oracle TimesTen In-Memory Database SQL Reference*. If the attribute is specified in an `ALTER SESSION` statement in a database where `PLSQL=0`, an error is returned. For example:

  ```
  ALTER SESSION SET PLSQL_OPTIMIZE_LEVEL = 2;
  ```

- 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.

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 TimesTen In-Memory Database PL/SQL Developer's Guide*. 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 TimesTen In-Memory Database PL/SQL Developer's Guide*. 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, described in *Oracle TimesTen In-Memory Database SQL Reference*. If this value is modified through `ALTER SESSION`, the new value impacts any PL/SQL program units that are currently running. For example:

```
ALTER SESSION SET PLSQL_TIMEOUT = 10;
```

Notes:

- If you are using PL/SQL, set the `PLSQL_TIIMEOUT` value to a value that is at least 5 seconds less than `TTC_TIMEOUT`.
- 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`.
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`.

See also

`TTC_Timeout`

TimesTen Cache first connection attributes

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

CacheAWTMethod

Determines whether asynchronous writethrough propagation uses the PL/SQL execution method or SQL array execution method to apply changes to the Oracle database server.

By default, asynchronous writethrough (AWT) uses PL/SQL execution method, `CacheAWTMethod=1`. AWT bundles all pending operations into a single PL/SQL collection that is sent to the Oracle database server to be executed. This method can improve AWT throughput when there are mixed transactions and network latency between TimesTen and the Oracle database server.

The SQL array execution to apply changes within TimesTen to the Oracle database 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 database server in one batch.

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. `1`(default) - 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. `1` (default) - Use PL/SQL collections and anonymous blocks (PL/SQL execution method).

TimesTen Cache database attributes

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

CacheAWTParallelism

CacheAWTParallelism indicates the number of threads that apply changes to the Oracle database. This attribute has a relationship to ReplicationParallelism and ReplicationApplyOrdering. The default is 1.

If you do not set this attribute or if you set it to the default value of 1, the number of threads that apply changes to the Oracle database is twice the setting for ReplicationParallelism. If the CacheAWTParallelism attribute is set to 1 or not set, the maximum allowable value for ReplicationParallelism is 16.

If both ReplicationParallelism and CacheAWTParallelism attributes are set, the value set in CacheAWTParallelism configures the number of threads used for parallel propagation. The setting for CacheAWTParallelism determines the number of apply threads for parallel propagation and the setting for ReplicationParallelism determines the number of threads for parallel replication.

To learn more about parallel AWT caching, see "Configuring parallel propagation to Oracle Database tables" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Required privilege

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

Setting

Set CacheAWTParallelism as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX odbc.ini file	CacheAWTParallelism	*n* - An integer between 1 and 31 that indicates the number of threads that apply changes to the Oracle database. The default is 1.
Windows ODBC Data Source Administrator	**Cache AWT Parallelism** field	*n* - An integer between 1 and 31 that indicates the number of threads that apply changes to the Oracle database. The default is 1.

CacheGridEnable

Enables or disables cache grid. 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` (default) - All cache groups in the database must be defined as members of a cache grid.
Windows ODBC Data Source Administrator	**Cache Grid Enable** check box	**unchecked** - Cache groups can be defined outside of a cache grid. **checked** (default) - All cache groups in the database must be defined as members of a cache grid.

CacheGridMsgWait

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

The maximum wait time includes the value of this attribute added to the value of `LockWait` connection attribute.

For more information on caching data from an Oracle database in a TimesTen cache grid, see *Oracle TimesTen Application-Tier 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`.
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`.

TimesTen Cache general connection attributes

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

DynamicLoadEnable

This attribute enables or disables dynamic load of data from an Oracle database to a TimesTen dynamic cache group. By default, dynamic load of data from an Oracle database 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 or using the statement level optimizer hint `TT_DynamicLoadEnable` in a SQL statement.

> **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 data from an Oracle database to TimesTen dynamic cache groups for the current connection. `1` (default) - Enables dynamic load of data from an Oracle database to TimesTen dynamic cache groups for the current connection.
Windows ODBC Data Source Administrator	**Dynamic Load Enable** field	`0` - Disables dynamic load of data from an Oracle database to TimesTen dynamic cache groups for the current connection. `1` (default) - Enables dynamic load of data from an Oracle database to TimesTen dynamic cache groups for the current connection.

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 data from an Oracle database in a TimesTen cache group, see *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Note: To override the value of this attribute at the statement level, set the `DynamicLoadErrorMode` optimizer flag with the `ttOptSetFlag` built-in procedure or using the statement level optimizer hint `TT_DynamicLoadErrorMode` in a SQL statement.

For details, see "Statement level optimizer hints" in the *Oracle TimesTen In-Memory Database SQL Reference*.

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` (default) - Statements execute against the cached data with no error. `1` - Statements use dynamic load or fail with an error.
Windows ODBC Data Source Administrator	**DynamicLoadErrorMode** field	`0` (default) - Statements execute against the cached data with no error. `1` - Statements use dynamic load or fail with an error.

OracleNetServiceName

The TimesTen Cache uses the `OracleNetServiceName` attribute.

This attribute identifies the Service Name for the Oracle instance.

To cache Oracle database tables and enable communication with the Oracle database, 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

The TimesTen Cache uses the `OraclePWD` attribute.

The value of this attribute is 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) 1996-2011, 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

PassThrough

The TimesTen Cache uses the `PassThrough` attribute.

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 TimesTen Cache, see *Oracle TimesTen Application-Tier 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 the Oracle database, 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 database connection, when using this attribute, see "Isolation" on page 1-61.

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` (default) - SQL statements are executed only on TimesTen. `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 (default) - SQL statements are executed only on TimesTen. 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.

Restrictions

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 database 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 the Oracle database.
- 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`, or `DELETE` operations depends on the setting of the `PassThrough` attribute as described in the table above. TimesTen 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 database 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 database applications, the changes are never reflected in TimesTen 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, passthrough execution of DDL statements does not result in commits on the TimesTen side.
- A transaction that contains operations that are replicated with `RETURN TWOSAFE` cannot have a `PassThrough` setting greater than `0`. If `PassThrough` is greater than `0`, an error is returned and the transaction must be rolled back.
- When `PassThrough` is set to `0`, `1`, or `2`, the following behavior occurs when a dynamic load condition exists:
  - A dynamic load can occur for a `SELECT` operation on cache tables in any dynamic cache group type.
  - A dynamic load for an `INSERT`, `UPDATE`, or `DELETE` operation can only occur on cached tables with dynamic asynchronous or synchronous writethrough cache groups.

Refer to "SQL Statements" in Oracle TimesTen In-Memory Database SQL Reference for details about the `INSERT`, `UPDATE`, `DELETE`, and `SELECT` statements.

RACCallback

This attribute enables 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 TimesTen Cache.

For more information about TimesTen Cache, see *Oracle TimesTen Application-Tier 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` (default) - Install the TAF and FAN callbacks.
Windows ODBC Data Source Administrator	**RACCallback** check box	**unchecked** - Do not install TAF and FAN callbacks. **checked** (default) - Install the TAF and FAN callbacks.

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, " TimesTen 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`

To view the value of client attributes, use the ODBC function `SQLGetConnectOption`. To learn more about this function see "Option support for SQLSetConnectOption and SQLGetConnectOption" section of the *Oracle TimesTen In-Memory Database C Developer's Guide*.

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 enables 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_DSN=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

TimesTen uses this attribute 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 enables 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`*`,`*`uppervalue`*.
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`*`,`*`uppervalue`*.

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 enables 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_DSN=Server_DSN;TCP_PORT=server_port"
```

or:

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

Once the logical server name is defined, you can use that name as the value for the `TTC_Server` attribute in a Client DSN. 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 nondefault port number, you must provide the port number in the ODBC connection string. For example:

```
"TTC_SERVER=server_host_name;TTC_SERVER_DSN=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 specifies 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 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.

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.

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 specifies 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 a network operation 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.

The operating systemselect() call on the client side of a CLIENT/SERVER connection uses the value of TTC_TIMEOUT. The SQLExecute() and OCIStmtExecute() functions do not use the value of this attribute.

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 active standby pair failover scenarios, the minimum value is 60 seconds.

If you are using PL/SQL, set the PLSQL_TIMEOUT value to a value that is at least 5 seconds less than TTC_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 following are true:

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

The timeout value can be set after establishing a connection by calling the ttIsql clienttimeout command. 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 the TimesTen Client waits for a connection before timing out. (The default value is 60.) In an active standby pair failover scenario, the minimum value is 60.
Windows ODBC Data Source Administrator	**Timeout Interval** field	A value between 0 and 99999 that represents the number of seconds that the TimesTen Client waits for a connection before timing out. (The default value is 60.) In an active standby pair failover scenario, the minimum 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 on a TimesTen Server system" in *Oracle TimesTen In-Memory Database Operations Guide*. Use these attributes 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 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, " TimesTen 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 ensure 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 -maxConnsPerServer 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 Chapter 4, "System Limits".

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 child server processes for a particular server DSN that will use round-robin connection distribution.

This attribute only has any effect if the TimesTen server is configured to operate in multithreaded mode (MaxConnsPerServer > 1). If ServersPerDSN is set to 1 then the first MaxConnsPerServer client connections to the server DSN will be assigned to one child server process, the next MaxConnsPerServer connections to a second child server process and so on.

If ServersPerDSN is set to N where N > 1 then the first N * MaxConnsPerServer client connections to the server DSN are distributed in round robin fashion over N child server processes. If additional client connections beyond N * MaxConnsPerServer are opened to the same server DSN then those connections are assigned to new child server processes sequentially as for the case when ServersPerDSN=1.

If this attribute is set in a server DSN definition then it will override, for that server DSN only, any value specified for the -serversPerDSN daemon option (specified in the ttendaemon.options file). If neither -serversPerDSN nor ServersPerDSN are configured then TimesTen uses the default value of 1. For further 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*.

Changes to TimesTen Server settings do not occur until the TimesTen server is restarted. To restart the Server, use the command ttDaemonAdmin -restartserver.

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 the minimum is 0 You generally should not need to set the serverstack size. However, if the ttcserver is getting repeatable core dumps, you may consider increasing the serverstacksize.
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. You generally should not need to set the serverstack size. However, if the ttcserver is getting repeatable Access Violations, you may consider increasing the serverstacksize.

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 `SQLExecDirect` call invokes the `ttOpsSetFlag` built-in procedure to tell 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. See the *Oracle TimesTen In-Memory Database PL/SQL Developer's Guide* for more details on this statement.

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 cannot be specified on a cache group with the `AUTOREFRESH` attribute, unless the cache group is dynamic. For explicitly loaded cache groups with the `AUTOREFRESH` attribute, 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 percent.
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 percent.
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, and so on. If the cycle is set to a value of `0`, aging occurs once every second.

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 percent, 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 TimesTen Application-Tier 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, and so on.

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, 2011-03-25 13:06:29.000000 >
< 2, 2011-03-25 13:06:42.000000 >
2 rows found.

Command> CALL ttAgingScheduleNow ('agingex');

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

See also

ttAgingLRUConfig
Oracle TimesTen Application-Tier Database Cache User's Guide

ttApplicationContext

Description

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

Result set

`ttApplicationContext` returns no results.

Examples

```
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.
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.

Column	Type	Description
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_BIGINT	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).

Examples

```
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_BIGINT	Last written offset in transaction log file.
forceLFN	TT_INTEGER	Last transaction log file forced to disk.
forceLFO	TT_BIGINT	Offset of last transaction log file forced to disk.
holdLFN	TT_INTEGER	Replication bookmark transaction log file.
holdLFO	TT_BIGINT	Replication bookmark log offset.

Examples

```
CALL ttBookmark ();
```

ttCacheAllowFlushAwtSet

Description

The `ttCacheAllowFlushAwtSet` built-in procedure enables you to execute a `FLUSH CACHE GROUP` statement against an AWT cache group and should only be used in a specific recovery scenario, as described in "When there is unsynchronized data in the cache groups" section in the *Oracle TimesTen In-Memory Database Replication Guide*.

Set auto commit to off before executing the `ttCacheAllowFlushAwtSet` built-in procedure when setting the *`enableFlush`* parameter to 1; otherwise, this parameter automatically resets to 0 directly after executing the built-in procedure. Then, perform a commit after you execute the `FLUSH CACHE GROUP` statement and execute the `ttCacheAllowFlushAwtSet` built-in procedure to reset the *`enableFlush`* parameter back to 0.

Required privilege

This procedure requires no privileges.

Syntax

```
ttCacheAllowFlushAwtSet (enableFlush)
```

Parameters

`ttCacheAllowFlushAwtSet` has the parameters:

Parameter	Type	Description
`enableFlush`	`TT_INTEGER`	`0` - The user is prevented from executing a `FLUSH CACHE GROUP` statement against an AWT cache group, which is the intended restriction. `1` - The user is allowed to execute a `FLUSH CACHE GROUP` statement against an AWT cache group, which should only be done for recovery, as described in "When there is unsynchronized data in the cache groups" section in the *Oracle TimesTen In-Memory Database Replication Guide*.

Result set

`ttCacheAllowFlushAwtSet` returns no results.

Examples

The following example shows how to execute the `ttCacheAllowFlushAwtSet` built-in procedure to first allow and then disallow a `FLUSH CACHE GROUP` statement to be executed against the `marketbasket` AWT cache group.

```
Command> set autocommit off;
Command> CALL ttCacheAllowFlushAwtSet(1);
Command> FLUSH CACHE GROUP marketbasket;
Command> CALL ttCacheAllowFlushAwtSet(0);
Command> COMMIT;
```

See also

"When there is unsynchronized data in the cache groups" section in the *Oracle TimesTen In-Memory Database Replication Guide*.

ttCacheAutorefIntervalStatsGet

Description

The `ttCacheAutorefIntervalStatsGet` built-in procedures returns statistical information about the last 10 autorefresh cycles for a particular autorefresh interval.

Required privilege

This procedure requires no privileges.

Syntax

`ttCacheAutorefIntervalStatsGet (`*`autoRefInterval`*`, `*`isStatic`*`)`

Parameters

`ttCacheAutorefIntervalStatsGet` has the parameters:

Parameter	Type	Description
autoRefInterval	TT_BIGINT NOT NULL	The *autorefreshInterval* designates the cache group (the one with this autorefresh interval value) on which to gather statistics. The integer value for the autorefresh interval (in milliseconds) is the same value that was originally specified when the autorefresh cache group was created to indicate how often autorefresh is scheduled.
isStatic	TT_INTEGER	Indicates if you are to retrieve information on static or dynamic cache groups with the interval value: 0 - dynamic cache groups 1 - static (non-dynamic) cache groups The default is static.

Result set

`ttCacheAutorefIntervalStatsGet` returns statistical information about the last 10 autorefresh cycles for a particular autorefresh interval:

Column	Type	Description
autorefInterval	TT_BIGINT	Autorefresh interval in milliseconds.
isStatic	TT_INTEGER	Indicates that the information is for static or dynamic cache groups with the interval value: 0 - dynamic cache groups 1 - static (non-dynamic) cache groups
autorefNumber	TT_BIGINT	Autorefresh number.
startTimestamp	TT_TIMESTAMP	Autorefresh start time.
selectLimit	TT_BIGINT	Select row limit set for incremental autorefresh cache group.
numRows	TT_BIGINT	Number of rows refreshed.

Column	Type	Description
numOps	TT_BIGINT	Number of SQL operations executed.
numCommits	TT_BIGINT	Number of commits.
commitBufSize	TT_BIGINT	Maximum commit buffer size in bytes.
commitBufMaxReached	TT_BIGINT	Amount of memory used for commit processing in bytes.
commitBufNumOverflows	TT_BIGINT	Number of times the commit buffer overflowed for each transaction.
totalNumRows	TT_BIGINT	Number of rows refreshed since the autorefresh thread was started.
totalNumOps	TT_BIGINT	Number of SQL operations were executed since the autorefresh thread was started.
totalNumCommits	TT_BIGINT	Number of commits since the autorefresh thread was started.
totalNumRollbacks	TT_BIGINT	Number of rollbacks since the autorefresh thread started
totalNumSnapshotOld	TT_BIGINT	Number of "Snapshot too old" errors received since the autorefresh thread started

Examples

The following example shows how to execute `ttCacheAutorefIntervalStatsGet` built-in procedure to retrieve statistics for autorefresh cache groups that have been defined as static and have the interval of seven seconds:

```
Command> call ttCacheAutorefIntervalStatsGet(7000,1);

< 7000, 1, 41, 2013-04-25 15:17:00.000000, 0, 0, 0, 1, 0, 0, <NULL>,
132121, 132121, 13, 21, 0, 0, 0, 0 >
< 7000, 1, 40, 2013-04-25 15:16:53.000000, 0, 0, 0, 1, 0, 0, <NULL>,
132121, 132121, 12, 21, 0, 0, 0, 0 >
< 7000, 1, 39, 2013-04-25 15:16:46.000000, 0, 0, 0, 1, 0, 0, <NULL>,
132121, 132121, 11, 21, 0, 0, 0, 0 >
< 7000, 1, 38, 2013-04-25 15:16:39.000000, 0, 0, 0, 1, 0, 0, <NULL>,
132121, 132121, 10, 21, 0, 0, 0, 0 >
< 7000, 1, 37, 2013-04-25 15:16:32.000000, 0, 6305, 6305, 1, 0, 131072,
<NULL>, 132121, 132121, 9, 21, 0, 0, 0, 0 >
< 7000, 1, 36, 2013-04-25 15:16:24.000000, 0, 15616, 15616, 1, 0, 131072,
<NULL>, 125816, 125816, 8, 21, 0, 0, 0, 0 >
< 7000, 1, 35, 2013-04-25 15:16:17.000000, 0, 18176, 18176, 1, 0, 131072,
<NULL>, 110200, 110200, 7, 21, 0, 0, 0, 0 >
< 7000, 1, 34, 2013-04-25 15:16:10.000000, 0, 14336, 14336, 1, 0, 131072,
<NULL>, 92024, 92024, 6, 21, 0, 0, 0, 0 >
< 7000, 1, 33, 2013-04-25 15:16:03.000000, 0, 15360, 15360, 1, 0, 131072,
<NULL>, 77688, 77688, 5, 21, 0, 0, 0, 0 >
< 7000, 1, 32, 2013-04-25 15:15:56.000000, 0, 11520, 11520, 1, 0, 131072,
<NULL>, 62328, 62328, 4, 21, 0, 0, 0, 0 >

10 rows found.
```

Notes

This procedure is available only for TimesTen Cache.

See also

ttCacheAutorefreshSelectLimit
ttCacheAutorefreshXactLimit
"Improving execution of large transactions when using incremental autorefresh for read-only cache groups" and "Configuring a select limit when using incremental autorefresh for read-only cache groups" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

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 ('cgOwner', 'cgName', synchronous)
```

Parameters

`ttCacheAutorefresh` has the parameters:

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

Examples

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 TimesTen Cache.

ttCacheAutorefreshLogDefrag

Description

The `ttCacheAutorefreshLogDefrag` built-in procedure compact the trigger log space for a cache autorefresh table.

For usage details, see "Defragmenting change log tables in the tablespace" in the *Oracle TimesTen Application-Tier Database Cache User's Guide.*

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheAutorefreshLogDefrag ('action')
```

Parameters

`ttCacheAutorefreshLogDefrag` has the parameters:

Parameter	Type	Description
action	`VARCHAR (50) NOT NULL`	Acceptable values are: `Compact` - Defragments only the trigger log space. `CompactAndReclaim` - Defragments the trigger log space and the transaction commit buffer (reclaim space). **NOTE:** The reclaim phase takes a lock on the trigger log table for a brief moment. This can suspend the workload from writing into the base table.

Result set

`ttCacheAutorefreshLogDefrag` returns no results.

Examples

In this example, the call compacts or defragments only the trigger log space.

```
Command> call ttCacheAutorefreshLogDefrag('Compact');
```

Notes

This procedure is available only for TimesTen Cache.

See also

`ttCacheConfig`
`ttCacheAutorefreshStatsGet`

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 ('cgOwner', 'cgname')
```

Parameters

ttCacheAutorefreshStatsGet has the parameters:

Parameter	Type	Description
cgOwner	VARCHAR2 (30)	Name of the cache group owner.
cgName	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" below.	Y
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" below.	Y

Column name	Column type	Description	Returned for full autorefresh
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 is 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 the number of rows updated on the Oracle database.	N
numOracleBytes	TT_BIGINT	The number of bytes transferred from the Oracle database 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 the Oracle database.	N
autorefQueryFetchDuration	TT_BIGINT	The duration in milliseconds that it takes for the autorefresh query to fetch rows from the Oracle database.	N
autorefTtApplyDuration	TT_BIGINT	The duration in milliseconds that it takes for TimesTen to apply the autorefresh.	N
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 the Oracle database. 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 the Oracle database 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

Column name	Column type	Description	Returned for full autorefresh
status	VARCHAR2 (128)	A string description of the status of the current autorefresh. See "Notes" below. Supported values for this field are: Complete inProgress Failed	Y
numlogrows	TT_BIGINT	Number of rows fetched from the Oracle database in this autorefresh.	Y
totalnumlogrows	TT_BIGINT	The cumulative number of rows fetched from the Oracle database in this autorefresh.	Y
autorefLogFragmentationPct	TT_BIGINT	A low-water mark for table usage by percentage. If less than the specified percent of the table is used, the table is compacted.	Y
autorefLogFragmentationTs	TT_TIMESTAMP	The timestamp when the last utilization/ fragmentation ratio was calculated	Y
autorefLogDefragGcnt	TT_BIGINT	The number of times the table has been compacted.	Y

Examples

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, 2011-07-23 15:43:52.000000, 850280, 44,
0, 75464, 528255, 75464, 310, 110, 6800, 1890912,
12439795, 1890912, 160020, InProgress, 2, 74 >
< 1164260, 2011-07-23 15:43:33.000000, 831700, 43,
13550, 108544, 759808, 108544, 1030, 230, 12290, 1815448,
11911540, 1815448, 160020, Complete, 2, 72 >
< 1164260, 2011-07-23 15:43:12.000000, 810230, 42,
17040, 115712, 809984, 115712, 610, 330, 16090, 1706904,
11151732, 1706904, 146470, Complete, 2, 70>
< 1164260, 2011-07-23 15:42:52.000000, 790190, 41,
14300, 94208, 659456, 94208,560, 320, 13410, 1591192,
10341748, 1591192, 129430, Complete, 2, 68 >
< 1164260, 2011-07-23 15:42:32.000000, 770180, 40,
12080, 99328, 695296, 99328,450, 290, 11340, 1496984,
9682292, 1496984, 115130, Complete, 2, 66 >
< 1164260, 2011-07-23 15:42:12.000000, 750130, 39,
10380, 86016, 598368, 86016,430, 230, 9720, 1397656,
8986996, 1397656, 103050, Complete, 2, 64 >
< 1164260, 2011-07-23 15:41:52.000000, 730130, 38,
13530, 112640, 700768, 112640, 530, 220, 12780, 1311640,
8388628, 1311640, 92670, Complete, 2, 62 >
< 1164260, 2011-07-23 15:41:32.000000, 710120, 37,
9370, 56320, 326810, 56320, 310, 160, 8900, 1199000,
7687860, 1199000, 79140, Complete, 2, 60 >
< 1164260, 2011-07-23 15:41:22.000000, 700120, 36,
2120, 10240, 50330, 10240, 50, 200, 1870, 1142680,
7361050, 1142680, 69770, Complete, 2, 58 >
```

```
< 1164260, 2011-07-23 15:41:12.000000, 690110, 35,
0, 0, 0, 0, 0, 0, 0, 1132440, 7310720, 1132440,
67650, Complete, 2, 56 >
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 TimesTen Cache.

ttCacheAutorefreshSelectLimit

Description

Configuring the incremental autorefresh to join the Oracle database base table with a limited number of rows from the autorefresh change log table is known as configuring a select limit. This is accomplished with the `ttCacheAutorefreshSelectLimit` built-in procedure.

Required privilege

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

Syntax

```
ttCacheAutorefreshSelectLimit ( autorefreshInterval, value )
```

Parameters

`ttCacheAutorefreshSelectLimit` has the parameters:

Parameter	Type	Description
autorefreshInterval	`TT_VARCHAR(30) NOT NULL`	The *autorefreshInterval* designates the cache group (the one with this autorefresh interval value) on which to apply the *value*. The integer value for the autorefresh interval (in milliseconds) is the same value that was originally specified when the autorefresh cache group was created to indicate how often autorefresh is scheduled.
value	`TT_VARCHAR(30)`	The *value* denotes a limit of the number of rows to select from the autorefresh change log file to apply to the cached table. These changes are applied incrementally until all the rows in the autorefresh change log table have been applied. If the value changes, it takes effect at the start of the next autorefresh cycle. The *value* can be one of the following: ■ '`ON`': Select at most 1000 rows at a time from the autorefresh change log table to apply for every autorefresh cycle. ■ *number*: Select at most a user specified number of rows from the autorefresh change log table during the autorefresh cycle. If the user specified a limit size of 2000 rows, then autorefresh selects at most 2000 rows at a time from the autorefresh change log table. If you specify a negative number, an error is returned. ■ '`OFF`': Disables the select limit. The incremental autorefresh selects all rows from the change log table during the autorefresh cycle. ■ `NULL`: If the *value* provided is `NULL` or not specified, the current setting is returned.

Result set

`ttCacheAutorefreshSelectLimit` returns the select limit value that has been set for a particular autorefresh interval:

Column	Type	Description
autorefreshInterval	TT_VARCHAR(30)	The *autorefreshInterval* that designates the cache group (the one with this autorefresh interval value).
value	TT_VARCHAR(30)	The current *value* that shows the number of rows that is selected from the autorefresh change log file to apply to the cached table.

Examples

You can show the current setting by either providing a NULL value or no parameter. The following example shows the setting for incremental autorefresh cache groups with an interval value of 7 seconds.

```
Command> call ttCacheAutorefreshSelectLimit('7000', NULL);
< 7000, 2000 >
1 row found.
Command> call ttCacheAutorefreshSelectLimit('7000');
< 7000, 2000 >
1 row found.
```

The following example set a select limit to 2000 rows for incremental autorefresh cache groups with an interval value of 7 seconds.

```
Command> call ttCacheAutorefreshSelectLimit('7000', '2000');
< 7000, 2000 >
1 row found.
```

Notes

- This procedure is available only for TimesTen Cache.
- The ttCacheAutotrefreshSelectLimit built-in procedure can set a select limit only on an interval that is defined for a single cache group that contains one table, where the cache group is defined as a static read-only cache group with incremental autorefresh.
- The setting for ttCacheAutorefreshSelectLimit is not replicated or duplicated. The user must execute the built-in on both the active and standby nodes.
- The settings do not reset if you drop all cache groups for the interval.
- The ttMigrate, ttBackup, and ttRestore built-in procedures do not preserve the setting of ttCacheAutorefreshSelectLimit.
- If you alter the cache group autorefresh interval, it does not modify what was set previously through execution of ttCacheAutorefreshSelectLimit for the cache group. You can only alter the select limit for the cache group with the ttCacheAutorefreshSelectLimit built-in procedure.

See also

ttCacheAutorefIntervalStatsGet
"Configuring a select limit when using incremental autorefresh for read-only cache groups" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

ttCacheAutorefreshXactLimit

Description

This procedure specifies the interval at which an immediate autorefresh on single table cache groups within a specified autorefresh interval and commits after the specified number of operations.

This procedure is useful if updates have occurred on the Oracle database and you want to refresh them on the cache group before the next scheduled autorefresh.

To modify the reclaim buffer size, use the `ttDBConfig` built-in procedure.

Required privilege

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

Syntax

```
ttCacheAutorefreshXactLimit ('IntervalValue', 'Value')
```

Parameters

`ttCacheAutorefreshXactLimit` has the parameters:

Parameter	Type	Description
intervalValue	VARCHAR2 (50) NOT NULL	Indicates the interval at which the autorefresh cache groups are defined to occur in units of milliseconds. *IntervalValue* is an integer value in milliseconds that was specified when the autorefresh cache group was created on how often autorefresh is scheduled.
value	VARCHAR2 (200)	The *Value* can be one of the following: ■ 'ON' - Enables autorefresh to commit after every 256 operations. ■ 'OFF' - Disables the transaction limit for autorefresh cache groups and sets autorefresh back to using a single transaction. ■ number - Denotes when to commit after a certain number of operations. For example, if the user specifies 1024, then autorefresh commits after every 1024 operations in the transaction. If you specify a negative number, an error is returned. ■ NULL - When the value is NULL, 0 or not specified, the current setting is returned.

Result set

`ttCacheAutorefreshXactLimit` returns the results:

Column	Type	Description
intervalValue	VARCHAR2 (50) NOT NULL	The interval at which the autorefresh cache groups are defined to occur in units of milliseconds.

Column	Type	Description
value	VARCHAR2 (200)	The *Value* can be one of the following: ■ 'ON' - Enables autorefresh to commit after every 256 operations. ■ 'OFF' - Disables the transaction limit for autorefresh cache groups and sets autorefresh back to using a single transaction. ■ *number* - Denotes when to commit after a certain number of operations. For example, if the user specifies 1024, then autorefresh commits after every 1024 operations in the transaction. If you specify a negative number, an error is returned. ■ NULL - When the value is NULL or not specified, the current setting is returned.

Examples

The following example sets up the transaction limit to commit after every 256 operations for all incremental autorefresh read-only cache groups that are defined with an interval value of 10 seconds.

```
call ttCacheAutorefreshXactLimit('10000', 'ON');
```

After the month end process has completed and the incremental autorefresh read-only cache groups are refreshed, disable the transaction limit for incremental autorefresh read-only cache groups that are defined with the interval value of 10 seconds.

```
call ttCacheAutorefreshXactLimit('10000', 'OFF');
```

To enable the transaction limit for incremental autorefresh read-only cache groups to commit after every 2000 operations, provide 2000 as the value as follows:

```
call ttCacheAutorefreshXactLimit('10000', '2000');
```

Notes

- This procedure is available only for TimesTen Cache. This built-in procedure only applies for static read-only cache groups with incremental autorefresh.
- While autorefresh is in-progress and is being applied in several small transactions, transactional consistency cannot be maintained. Once the autorefresh cycle has completed, the data is transactional consistent.
- The setting for ttCacheAutorefreshXactLimit is not replicated or duplicated. The user must execute the built-in procedure on both the active and standby nodes.
- The settings do not reset if you drop all cache groups for the interval.
- The ttMigrate, ttBackup, and ttRestore built-in procedures do not preserve the setting of ttCacheAutorefreshXactLimit.
- If you alter the cache group autorefresh interval, it does not modify the setting of ttCacheAutorefreshXactLimit.

See also

ttCacheAutorefIntervalStatsGet

"Improving execution of large transactions when using incremental autorefresh for read-only cache groups" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

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.

Run this procedure on the replication node that is replicating AWT changes to the Oracle database. If the active standby pair is functioning normally, the node replicating AWT changes is the standby. If the active is operating standalone, the node replicating AWT changes is the active.

If a failure occurs on the node where the active database resides, the standby node becomes the new active node. In that case you would run this procedure on the new active node.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

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

Parameters

`ttCacheAWTMonitorConfig` has the optional 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 *samplingRate* 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 *samplingRate* is 16. If state is set to `OFF`, the default for *samplingRate* 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`.

Column	Type	Description
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 3-102

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 the result is `0`, there is no set limit.

Examples

```
CALL ttCacheAWTThresholdGet();
```

Notes

This procedure is available only for TimesTen 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 the Oracle database. 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.

Examples

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

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

Notes

This procedure is available for TimesTen Cache.

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

See also

`ttCacheAWTThresholdGet`

ttCacheCheck

Description

The `ttCacheCheck` built-in procedure performs a check for missing constraints for cached tables on the Oracle database.

Any unique index, unique constraint, or foreign key constraint on columns in Oracle Database tables that are to be cached should also be created on asynchronous writethrough cache tables within TimesTen. If you have not created these constraints on the AWT cache tables and you have configured the cache group for parallel propagation, TimesTen serializes any transactions with DML operations to those tables with missing constraints.

This procedure provides information about missing constraints and the tables marked for serialized propagation.

Call `ttCacheCheck` to manually check for missing constraints, under these conditions:

- After completing a series of `DROP CACHE GROUP` statements.
- After creating or dropping a unique index or foreign key on the Oracle database.
- To determine why some transactions are being serialized.

This procedure updates system tables to indicate if DML executed against a table should or should not be serialized, therefore you must commit or roll back after the `ttCacheCheck` built-in completes.

For more details on parallel propagation, see "Configuring parallel propagation to Oracle Database tables" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Required privilege

This procedure requires the `CACHE_MANAGER` privilege.

Syntax

```
ttCacheCheck('operation', cgOwner, cgName)
```

Parameters

`ttCacheCheck` has these parameters:

Parameter	Type	Description
operation	`TT_VARCHAR(30)`	Specifies the constraint to be checked. Legal values are: ■ `ForeignKey` -Checks foreign key constraints ■ `Unique` - Checks unique constraints ■ `Awt` - Checks both foreign key and unique constraints ■ `NULL` - Checks both foreign key and unique constraints

Parameter	Type	Description
cgOwner	TT_VARCHAR(30)	Specifies the owner of the cached Oracle database table. If NULL, checks all asynchronous writethrough cache groups owned by the connection user. If both *cgOwner* and *cgName* are NULL, checks all asynchronous cache groups.
cgName	TT_VARCHAR(30)	Specifies the name of the cached Oracle database table. If NULL, but the *cgOwner* is specified checks all asynchronous writethrough cache groups owned by *cgOwner*. If both *cgOwner* and *cgName* are NULL, checks all asynchronous cache groups.

Result set

ttCacheCheck returns the result set:

Column	Type	Value
cgOwner	TT_VARCHAR(30) NOT NULL	The owner of the cache group.
cgName	TT_VARCHAR(30) NOT NULL	The name of the cache group.
tblOwner	TT_VARCHAR(30)	The owner of the table.
tblName	TT_VARCHAR(30)	The name of the table.
objectType	TT_VARCHAR(15)	The type of Oracle object: unique index, constraint or foreign key.
objectOwner	TT_VARCHAR(30)	The owner of the Oracle object.
objectName	TT_VARCHAR(30)	The object name.
msgType	TT_SMALLINT NOT NULL	The type of message: 0 = Informational 1 = Warning -1 = Error
msg	TT_VARCHAR(100000) NOT NULL	Message describing the issue.
objectDesc	VARCHAR2(200000)	A description of the object. If the object is AWT checking, the description is the SQL statement that describes the object.

Examples

The following example determines if there are any missing constraints for the cache group update_orders that is owned by cacheuser. A result set is returned that includes the warning message. The ordertab table in the update_orders cache group is marked for serially propagated transactions.

```
Command> call ttCacheCheck( NULL, 'cacheuser', 'update_orders');

< CACHEUSER, UPDATE_ORDERS, CACHEUSER, ORDERTAB, Foreign Key, CACHEUSER,
CUST_FK, 1, Transactions updating this table will be serialized to Oracle
because: The missing foreign key connects two AWT cache groups.,
table CACHEUSER.ORDERTAB constraint CACHEUSER.CUST_FK foreign key(CUSTID)
```

```
references CACHEUSER.ACTIVE_CUSTOMER(CUSTID) >
1 row found.
```

Notes

This procedure is available only for TimesTen Cache.

See also

ttCacheDbCgStatus
ttCachePolicyGet
ttCachePolicySet
ttCacheStart
ttCacheStop
ttCacheUidGet
ttCacheUidPwdSet
"ttAdmin" on page 3-3

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 database 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*: ■ AgentFailoverTimeout - When working in an Oracle RAC environment, sets the TAF timeout, in minutes. Configures how long TAF retries when establishing a connection. The default is four minutes. ■ AgentTimeout - Number of seconds before a database is declared terminated if the cache agent cannot connect to the Oracle database server. ■ AutoRefreshLogFragmentationWarningPCT - The percent of table usage that must occur before warning the user to compact the table. ■ AutoRefreshLogDeFragmentAction - Compaction mode for the specified tables. ■ AutoRefreshLogTblSpaceUsagePCT - Specifies the cache administration user's tablespace usage warning threshold as a percentage. ■ DeadDbRecovery - Specifies the type of autorefresh recovery when the cache agent restarts. ■ TblSpaceFullRecovery - Specifies the action that TimesTen takes when the cached Oracle database table is updated and the cache administration user's tablespace is full.
tblOwner	VARCHAR2(30)	Specifies the owner of the cached Oracle database 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 database 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 AgentFailoverTimeout, it specifies the number of minutes before TAF retries when establishing a connection, when working in an Oracle RAC environment. The default is four minutes. ■ 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 database server. The default is 0, which means that the database is never declared terminated. ■ When *Param* is AutoRefreshLogFragmentationWarningPCT, the value of the fourth parameter must be an integer between 1 and 100, representing a percentage of the table. ■ When *Param* is AutoRefreshLogDeFragmentAction, the value can be Manual, CompactOnly or CompactandReclaim If Manual is specified no action is taken. The user can run ttCacheAutorefreshLogDefrag built-in procedure to defragment the logs. If CompactOnly is specified trigger log space is compacted. If CompactandReclaim is specified both the trigger log space and the transaction log buffer (reclaim space) are compacted. The default is Manual. ■ When *Param* is AutoRefreshLogTblSpaceUsagePCT, the value can be 0 to 100. The default is 0, which means no warning is returned regardless of the tablespace usage. ■ 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 database error is returned when the cached Oracle database table is updated. The default is None. Or Specifies the value to be set by *AwtErrorXmlOutput*: ■ ASCII - A text file that contains the AWT error report. (Default) ■ XML - An XML file that contains the AWT error report and the associated DTD file.

Result set

ttCacheConfig returns no results when an application uses it to set parameter values. When it is used to return parameter settings, ttCacheConfig returns the following results.

Column	Type	Value
Param	VARCHAR2(50)	Parameter name: AgentTimeout AgentFailoverTimeout AutoRefreshLogTblSpaceUsagePCT DeadDbRecovery TblSpaceFullRecovery
tblOwner	VARCHAR2(30)	Owner of the cached Oracle database table.
tblName	VARCHAR2(30)	Name of the cached Oracle database 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 database server. ■ When *Param* is AutoRefreshLogTblSpaceUsagePCT, the value can be 0 to 100. ■ When *Param* is DeadDbRecovery, the value can be Normal or Manual. ■ When *Param* is TblSpaceFullRecovery, the value can be Reload or None.

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

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

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

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

When working in an Oracle RAC environment, the following shows how to retrieve the value of the failover timeout:

```
Command> call ttCacheConfig('AgentFailoverTimeout');
< AgentFailoverTimeout, <NULL>, <NULL>, 4 >
1 row found.
.
```

The following sets the failover timeout to 5 minutes:

```
Command> call ttCacheConfig('AgentFailoverTimeout',,,5);
< AgentFailoverTimeout, <NULL>, <NULL>, 5 >
1 row found.
Command>
```

Notes

This procedure is available only for TimesTen Cache.

You must call the `ttCacheConfig` built-in procedure from every node in a cache grid or a active standby pair.

See also

ttCacheDbCgStatus
ttCachePolicyGet
ttCachePolicySet
ttCacheStart
ttCacheStop
ttCacheUidGet
ttCacheUidPwdSet
"ttAdmin" on page 3-3
"Reporting Oracle Database permanent errors for AWT cache groups", "Managing a Caching Environment", and "Setting up TimesTen Cache in an Oracle RAC environment" (regarding Agent Failover) in *Oracle TimesTen Application-Tier Database Cache User's Guide*

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 autorefresh status of all the cache groups in the database. 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 database server. `dead` - The cache agent was not able to contact the Oracle database 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 database server. The status of at least one cache group is `recovering`.
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 database server. `dead` - The cache agent was not able to contact the Oracle database within the timeout period and the specified cache group may be out of sync with the Oracle database server. `recovering` - The specified cache group is being resynchronized with the Oracle database 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.
```

Notes

This procedure is available only for TimesTen Cache.

See also

ttCacheConfig
ttCachePolicyGet
ttCachePolicySet
ttCacheStart
ttCacheStop
ttCacheUidGet
ttCacheUidPwdSet
"ttAdmin" on page 3-3

ttCacheDDLTrackingConfig

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

DDL tracking saves the change history for all the cached Oracle database tables. One DDL tracking table is created to store DDL statements issued on any cached Oracle database table. You can use this information to diagnose autorefresh problems.

See "Tracking DDL statements issued on cached Oracle Database tables" in *Oracle TimesTen Application-Tier 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 database tables are tracked. Valid values are: `enable` - Enables tracking. `disable` (default) - Disables tracking.

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` (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 ();
```

Notes

This procedure is available only for TimesTen Cache.

See also

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

ttCacheUidGet
ttCacheUidPwdSet
"ttAdmin" on page 3-3

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` (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 TimesTen 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" on page 3-3

ttCachePropagateFlagSet

Description

This procedure enables you to disable propagation of committed updates (the result of executing DML statements) within the current transaction to the Oracle database. Any updates from executing DML statements after the flag is set to zero are never propagated to the back-end Oracle database. Thus, these updates exist only on the TimesTen database. You can then re-enable propagation for DML statements by resetting the flag.

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 the Oracle database. The flag remains set until the end of the transaction or until the procedure is set to `1`. If `1`, updates are sent to the Oracle database.

Result set

`ttCachePropagateFlagSet` returns no results.

Notes

This procedure is available only for TimesTen Cache.

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

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

The propagate flag is reset after a commit or rollback.

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

Examples

This example sets `autocommit` off to prevent the propagation flag from toggling from off to on after a commit. Calls the `ttCachePropagateFlagSet` to turn off propagation. A row is inserted into the TimesTen Cache detail table for `oratt.writetab`. Then, propagation is reenabled by calling the `ttCachePropagateFlagSet` built-in procedure and setting the flag to one.

```
Command> set autocommit off;
Command> call ttCachePropagateFlagSet(0);
Command> INSERT INTO oratt.writetab VALUES (103, 'Agent');
1 row inserted.
Command> COMMIT;
Command> SELECT * FROM oratt.writetab;
< 100, Oracle >
< 101, TimesTen >
< 102, Cache >
< 103, Agent >
4 rows found.
Command> call ttCachePropagateFlagSet(1);
```

When you select all rows on the Oracle database, the row inserted when propagation was turned off is not present in the `oratt.writetab` table on Oracle.

```
Command> set passthrough 3;
Command> SELECT * FROM oratt.writetab;
< 100, Oracle >
< 101, TimesTen >
< 102, Cache >
3 rows found.
```

ttCacheSqlGet

Description

This procedure generates the Oracle SQL statements to install or uninstall Oracle database 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 database 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 database 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 database 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 database objects.

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

Examples

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

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

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

Notes

This procedure is available only for TimesTen Cache.

Each returned *retval* field contains a separate Oracle SQL statement that may be directly executed on the Oracle database. 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 database 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 TimesTen Cache.

The cache administration user ID and password must be set with the `ttCacheUidPwdSet` built-in procedure 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" on page 3-3

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 stop the cache agent if it does not stop within *timeout* seconds. If set to `0`, the daemon waits forever for the cache agent. The default value is `100`.

Result set

`ttCacheStop` returns no results.

Examples

To stop the TimesTen cache agent, use:

```
CALL ttCacheStop();
```

Notes

This procedure is available only for TimesTen 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 database 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" on page 3-3

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 with the `ttCacheUidPwdSet` built-in procedure, `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.

Examples

```
CALL ttCacheUidGet();
```

Notes

This procedure is available only for TimesTen Cache.

See also

`ttCacheUidPwdSet`
"ttAdmin" on page 3-3

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.

Examples

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

Notes

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

This procedure is available only for TimesTen Cache.

For all levels of `DDLReplicationLevel`, you can set the cache administration user ID and password while the cache or replication agents are running. For more details on changing the cache administration user ID or password, see "Changing cache user names and passwords" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

The cache administration user ID cannot be reset while there are cache groups on the database. The cache administration password can be changed at any time.

See also

`ttCacheUidGet`
`ttCacheUidPwdSet`
"ttAdmin" on page 3-3

ttCkpt

Description

This procedure performs a nonblocking checkpoint operation. For information on blocking checkpoints, see "ttCkptBlocking" on page 2-56. A checkpoint operation makes a record of the current state of the database on disk and to purge transaction log files. A nonblocking 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 nonblocking checkpoints, TimesTen uses the log to recover.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttCkpt([timeout], [retries])
```

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, 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.

Examples

```
CALL ttCkpt();
```

Notes

For a description of checkpoints, see "Transaction Management" 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 makes 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 should 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, use this blocking checkpoint with caution. To perform a nonblocking 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, 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.

Examples

```
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" 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 the following 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 MB.
`ckptRate`	`TT_INTEGER NOT NULL`	Current rate at which TimesTen writes checkpoints 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(0, 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 MB (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` - nonblocking checkpoint. The background checkpointer performs this type if possible. `None` - For temporary databases, which have no checkpoint files.
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.

Column	Type	Description
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_BIGINT	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*.
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. The value is non-NULL if you call this procedure while a checkpoint is in progress.

Column	Type	Description
ckptVNo	TT_INTEGER NOT NULL	The checkpoint sequence number that is incremented for each checkpoint.

Examples

This example shows a checkpoint in progress:

```
Call ttckpthistory;
< 2011-04-14 16:56:34.169520, <NULL>, Fuzzy           ,
In Progress     , User
, <NULL>, 0, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>,
<NULL>, <NULL>, <NULL>, 13, 6 >

< 2011-04-14 16:55:47.703199, 2011-04-14 16:55:48.188764,
Fuzzy           , Completed       , Checkpointer    , <NULL>,
1, 0, 8964304, 294, 33554432, 291, 5677288, 27, 1019512,
1065408, <NULL>, 5 >

< 2011-04-14 16:54:47.106110, 2011-04-14 16:54:47.723379,
Static          , Completed       , Subdaemon       , <NULL>,
0, 0, 8960328, 294, 33554432, 291, 5677288, 256, 33157172,
5321548, <NULL>, 4 >

< 2011-04-14 16:54:41.633792, 2011-04-14 16:54:42.568469,
Blocking        , Completed       , User            , <NULL>,
1, 0, 8958160, 294, 33554432, 291, 5677288, 31, 1162112,
6604976, <NULL>, 3 >

< 2011-04-14 16:54:37.438827, 2011-04-14 16:54:37.977301,
Static          , Completed       , User            , <NULL>,
0, 0, 1611984, 93, 33554432, 92, 1853848, 93, 33554432,
1854052, <NULL>, 2 >

< 2011-04-14 16:54:36.861728, 2011-04-14 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;
< 2011-04-14 16:57:14.476860, 2011-04-14 16:57:14.477957,
Fuzzy           , Failed , User            , 847, 1, <NULL>,
<NULL>, 0, 0, 0, 0, 0, 0, 0, <NULL>, 7 >

< 2011-04-14 16:56:34.169520, 2011-04-14 16:56:59.715451,
Fuzzy           , Completed       , User            , <NULL>,
0, 0, 8966472, 294, 33554432, 291, 5677288, 5, 522000,
532928, <NULL>, 6 >

< 2011-04-14 16:55:47.703199, 2011-04-14 16:55:48.188764,
Fuzzy           , Completed       , Checkpointer    , <NULL>,
1, 0, 8964304, 294, 33554432, 291, 5677288, 27, 1019512,
1065408, <NULL>, 5 >

< 2011-04-14 16:54:47.106110, 2011-04-14 16:54:47.723379,
Static          , Completed       , Subdaemon       , <NULL>,
```

```
0, 0, 8960328, 294, 33554432, 291, 5677288, 256, 33157172,
5321548, <NULL>, 4 >

< 2011-04-14 16:54:41.633792, 2011-04-14 16:54:42.568469,
Blocking        , Completed       , User            , <NULL>,
1, 0, 8958160, 294, 33554432, 291, 5677288, 31, 1162112,
6604976, <NULL>, 3 >

< 2011-04-14 16:54:37.438827, 2011-04-14 16:54:37.977301,
Static          , Completed       , User            , <NULL>,
0, 0, 1611984, 93, 33554432, 92, 1853848, 93, 33554432,
1854052, <NULL>, 2 >

< 2011-04-14 16:54:36.861728, 2011-04-14 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

ttCommitBufferStats

Description

This built-in procedure returns the number of commit buffer overflows and the high watermark for memory used by transaction reclaim records during transaction commit process.

The information provided by the results of this procedure call is useful information when you want to explicitly set the maximum size of commit buffer, using the `CommitBufferSizeMax` connection attribute or the `ALTER SESSION` SQL statement, described in *Oracle TimesTen In-Memory Database SQL Reference*. This procedure helps you choose the right size for the reclaim buffer, based on the number of overflows and the maximum memory used by the reclaim records.

If there are buffer overflows, you may consider increasing the commit buffer maximum size. If there are no overflows and the highest amount of memory usage is well under the commit buffer maximum size, you may consider decreasing the maximum size.

For more information on reclaim operations, including details about setting the commit buffer size, see "Transaction reclaim operations" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires no privileges.

Syntax

```
ttCommitBufferStats( )
```

Parameters

`ttCommitBufferStats` takes no parameters.

Result set

`ttCommitBufferStats` returns these results:

Column	Type	Description
overflows	`TT_BIGINT NOT NULL`	Total number of commit buffer overflows.
maxReached	`TT_BIGINT NOT NULL`	The currently used maximum for the transaction commit buffer in bytes.

Examples

This shows the result for a session where there have been no commit buffer overflows and the transaction commit buffer is set to 500 MB.

```
Command> ALTER SESSION SET COMMIT_BUFFER_SIZE_MAX = 500;
Session altered.
Command> CALL ttCommitBufferStats( );
< 0, 524288000 >
1 row found
```

For a session where there have been 10 commit buffer overflows and the transaction commit buffer is set to 2 MB, the output of this procedure is:

```
Command> ALTER SESSION SET COMMIT_BUFFER_SIZE_MAX = 2;
Session altered.
Command> CALL ttCommitBufferStats( );
< 0, 2097152 >
1 row found
```

Notes

When you call the built-in procedure `ttCommitBufferStatsReset`, the commit buffer statistics are expressed in bytes. However, the `ttConfiguration` output and the value set by the connection attribute `CommitBufferSizeMax` are expressed in MB.

See also

`ttCommitBufferStatsReset`

ttCommitBufferStatsReset

Description

The `ttCommitBufferStatsReset` procedure resets transaction commit buffer statistics to `0`. This is useful, for example, if you have set a new value for the commit buffer maximum size and want to restart the statistics.

For more information on reclaim operations, including details about setting the commit buffer size, see "Transaction reclaim operations" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires no privileges.

Syntax

```
ttCommitBufferStatsReset()
```

Parameters

`ttCommitBufferStatsReset` takes no parameters.

Result set

`ttCommitBufferStatsReset` returns no result set.

Examples

```
CALL ttCommitBufferStatsReset;
```

See also

`ttCommitBufferStats`

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.

Examples

```
CALL ttCompact;
```

Notes

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.

Examples

```
CALL ttCompactTS (5);
```

Notes

Compacting data does not modify result addresses.

See also

ttCompact

ttComputeTabSizes

Description

The `ttComputeTabSizes` built-in procedure refreshes table size statistics stored in TimesTen system tables. After calling this built-in procedure, you can review the statistics updates by querying the `DBA_TAB_SIZES`, `USER_TAB_SIZES` or `ALL_TAB_SIZES` view.

This procedure computes the different types of storage allocated for the specified table, such as the amount of storage allocated for inline row storage, out-of-line buffers and system usage. If no table is specified, the procedure computes the sizes for all tables on which the user has `SELECT` privileges.

The execution of this built-in behaves like a DDL statement: the transaction commits just before the procedure begins and commits again upon its successful termination.

Required privilege

This procedure requires the `SELECT` privilege on the specified table.

Syntax

```
ttComputeTabSizes (['tblName'], [includeOutOfLine])
```

Parameters

`ttComputeTabSizes` has the parameters:

Parameter	Type	Description
tblName	`TT_CHAR(61)`	Name of an application table. Can include the table owner. If a value of `NULL` or an empty string is provided, updates the statistics for all the current tables. The type of tables that can be estimated are: ■ User tables, including cache group tables ■ Materialized views ■ System tables
includeOutOfLine	`TT_INTEGER`	`0` (no) or `1` (yes). Default is `1` (yes). If value is `0` (no), the procedure does not compute the size of out-of-line values for any table that has out-of-line columns. The out-of-line fields are displayed as `NULL`. Avoiding the computation of out-of-line values significantly decreases the latency of this procedure.

Result set

`ttComputeTabSizes` returns no results.

Examples

To compute the size of `my_table` without including out-of-line columns, use:

```
CALL ttComputeTabSizes ('my_table', 0);
```

Notes

The built-in procedure enables concurrent insertions while `ttComputeTabSizes` is executing. For this reason, the size computed by `ttComputeTabSizes` for each table is any value between the minimum size of the table during the computation and the maximum size of the table during the computation. For example, if the size of a table is 250 MB when `ttComputeTabSizes` is executed, and a transaction running concurrently raises the size of the table to 300 MB, `ttComputeTabSizes` estimates a value between 250 and 300 MB.

See also

`ttSize`

ttConfiguration

Description

The `ttConfiguration` built-in procedure returns the values for most, but not all, connection attributes for the current database connection. Specifically, the `ttConfiguration` built-in procedure returns the values for these connection attributes:

```
CacheAwtMethod
CacheAwtParallelism
CacheGridEnable
CacheGridMsgWait
CkptFrequency
CkptLogVolume
CkptRate
CkptReadThreads
CommitBufferSizeMax
ConnectionCharacterSet
ConnectionName
Connections
DDLCommitBehavior
DDLReplicationAction
DDLReplicationLevel
DataBaseCharacterSet
DataStore
DynamicLoadEnable
DuplicateBindMode
DurableCommits
DynamicLoadErrorMode
Isolation
RangeIndexType
LockLevel
LockWait
LogAutoTruncate
LogBufMB
LogBufParallelism
LogDir
LogFileSize
LogFlushMethod
LogPurge
MemoryLock
NLS_LENGTH_SEMANTICS
NLS_NCHAR_CONV_EXCP
NLS_SORT
OracleNetServiceName
PLSCOPE_SETTINGS
PLSQL
PLSQL_CCFLAGS
PLSQL_CODE_TYPE
PLSQL_CONN_MEM_LIMIT
PLSQL_MEMORY_ADDRESS
PLSQL_MEMORY_SIZE
PLSQL_OPTIMIZE_LEVEL
PLSQL_TIMEOUT
PassThrough
```

```
PermSize
PermWarnThreshold
Preallocate
PrivateCommands
QueryThreshold
RACCallback
ReceiverThreads
RecoveryThreads
ReplicationApplyOrdering
ReplicationParallelism
ReplicationTrack
SQLQueryTimeout
TempSize
TempWarnThreshold
Temporary
TypeMode
UID
```

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.

Examples

To see the value of the `QueryThreshold` connection 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 >
. . .
```

Notes

The values of client driver attributes are not returned by this procedure.

The values of other attributes, such as `ForceConnect`, may not be returned by this procedure, as well.

See also

Chapter 1, "Connection Attributes"

ttContext

Description

This procedure returns the context value of the current connection as a `BINARY(8)` value. You can use the context 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 context value.

Examples

```
CALL ttContext;
```

Notes

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" on page 3-140

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.

Examples

```
CALL ttDataStoreStatus('/data/Purchasing');
```

See also

`ttContext`
"ttStatus" on page 3-140

ttDBConfig

Description

The `ttDBConfig` built-in enables users to set or view the value of a TimesTen database system parameter.

Required privilege

This procedure requires `ADMIN` privilege.

Syntax

```
ttDBConfig('param', 'value')
```

Parameters

`ttDBConfig` has the parameters:

Parameter	Type	Description

param	`VARCHAR2(30) NOT NULL`	.A system parameter for which you either want to set a value or to see the current value. Accepted values for this argument are:
		`CacheParAwtBatchSize`
		`CacheAwtMethod`
		`CacheAgentCommitBufSize`
		`ParReplMaxDrift`
		`RepAgentCommitBufSize`

value	`VARCHAR2(200)`	The value for the system parameter. If you do not specify a value, this procedure returns the current value of the specified parameter.

Parameter / Value Pairs

These name/value pairs can be returned in the result set:

Name	Value	Description
`CacheParAwtBatchSize`	Number of rows in a batch	Configures a threshold value for the number of rows included in a single batch. Once the maximum number of rows is reached, TimesTen includes the rest of the rows in the transaction (TimesTen does not break up any transactions), but does not add any more transactions to the batch. **NOTE:** You should not change the value of this parameter unless advised by Oracle TimesTen technical support.

Name	Value	Description
CacheAwtMethod	0 - SQL Array execution method 1 - PL/SQL Execution method	Determines whether PL/SQL execution method or SQL array execution method is used for AWT propagation to apply changes to the Oracle database server. See the description of the CacheAWTMethod connection attribute for details. If set with this built-in procedure, overrides the value of the value of the connection attribute value.
CacheAgentCommitBufSize	Size in MB	Specifies the reclaim buffer maximum size for the cache agent. The cache agent periodically checks to see if the value has changed. The size cannot be greater than the temporary partition size. For more details, see "Improving performance when reclaiming memory during autorefresh operations" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.
ParReplMaxDrift	Number of seconds	Specifies the number of seconds of drift to allow between the parallel replication tracks. When you use automatic parallel replication and disable commit dependencies, some tracks may move ahead of the others. Once this threshold is passed, TimesTen synchronizes all replication tracks so that they catch up to each other. By default, this is set to zero, which means that checking for drift between tracks is disabled. For more details, see "Configuring automatic parallel replication with disabled commit dependencies" in the *Oracle TimesTen In-Memory Database Replication Guide*
RepAgentCommitBufSize	Size in MB	Specifies the reclaim buffer maximum size for the replication agent. The replication agent periodically checks to see if the value has changed. The size cannot be greater than the temporary partition size. For more details, see "Improving performance when reclaiming memory during autorefresh operations" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Result set

ttDBConfig returns the result set:

Column	Type	Description
param	VARCHAR2(30)	The name of the system parameter.
value	VARCHAR2(200)	Displays the current value of the specified parameter.

Examples

To retrieve the current value of the `CacheParAwtBatchSize`, use:

```
CALL ttDBConfig('CacheParAwtBatchSize');
<CACHEPARAWTBATCHSIZE, 125>
1 row found.
```

To set the value of the `RepAgentCommitBufSize` to 50 MB, use:

```
CALL ttDBConfig('RepAgentCommitBufSize', '50');
 <REPAGENTCOMMITBUFSIZE, 50>
 1 row found.
```

To set the current value of the `CacheAgentCommitBufSize` to 100, use:

```
Command> call ttDBConfig('CacheAgentCommitBufSize', '100');
 < CACHEAGENTCOMMITBUFSIZE, 100 >
1 row found.
```

Notes

After using this built-in procedure to set a parameter value, initiate a checkpoint to ensure the persistence of the parameter change. See details about the `ttCheckpoint` procedure in "Checkpoint operations" in the *Oracle TimesTen In-Memory Database Operations Guide*. For details about the checkpoint built-in procedure, see "ttCkpt" in this chapter.

Changes to parameter values made by `ttDbConfig` cannot be rolled back.

See also

"CacheAWTMethod" on page 1-97
"Improving AWT throughput", "Configuring batch size for parallel propagation for AWT cache groups", and "Improving performance when reclaiming memory during autorefresh operations" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*

ttDbWriteConcurrencyModeGet

Description

The `ttDbWriteConcurrencyModeGet` built-in returns information about the write concurrency mode of the database and the status of write concurrency mode operations and transitions.

Required privilege

This procedure requires no privilege.

Syntax

```
ttDbWriteConcurrencyModeGet()
```

Parameters

`ttDbWriteConcurrencyModeGet` has no parameters:

Result set

`ttDbWriteConcurrencyModeGet` returns the result set:

Column	Type	Description
ts	`TIMESTAMP NOT NULL`	Time at which the status information was collected.
mode	`TT_INTEGER NOT NULL`	The write concurrency mode: `0` - Optimize according to hints and standard optimization techniques. `1`- Optimize for concurrent write operations.
operation	`VARCHAR2 (50)`	The transition status of the write concurrency mode. Either: `NULL` - Not in transition. `TRANSITIONING TO MODE=`*n* where *n*= `0` or `1`.
status	`VARCHAR2 (100) NOT NULL`	The status of the write concurrency mode transition. Either: `IN TRANSITION` or `COMPLETE`.
msg	`VARCHAR2 (5000)`	`NULL` or a status explanation message.

Examples

The following example shows how to determine if your database is optimized for concurrent write operations:

```
Command> CALL ttDbWriteConcurrencyModeGet();

< 2013-09-23 13:48:21.207599, 1, <NULL>, COMPLETE, <NULL> >
1 row found.
```

The results indicate that at approximately 1:48 pm on September 23, 2013 the database was optimized for concurrent write operations. The mode was not in transition.

See also

ttDbWriteConcurrencyModeSet

ttDbWriteConcurrencyModeSet

Description

The ttDbWriteConcurrencyModeSet built-in enables control over read optimization during periods of concurrent write operations.

Set the mode to one (1) to enable the enhanced write concurrency mode and disable read optimization. Set the mode to zero (0) to disable the enhanced write concurrency mode and re-enable read optimization.

When the mode is set to one (1), all transaction and statement table lock hints are suppressed. This affects hint-triggered Sn table locks for SELECT statements and subqueries and also hint-triggered W table locks for DML statements. Suppression of the table lock hint also suppresses other table-lock hint driven execution plans such as star joins. Regardless of the mode setting, table locks that are not triggered by table-lock hints are not affected.

Required privilege

This procedure requires ADMIN privilege.

Syntax

```
ttDbWriteConcurrencyModeSet(mode, wait)
```

Parameters

ttDbWriteConcurrencyModeSet has these parameters:

Parameter	Type	Description
mode	TT_INTEGER NOT NULL	The write concurrency mode: 0 - Optimize according to hints and standard optimization techniques. 1 - Optimize for concurrent write operations.
wait	TT_INTEGER NOT NULL	0 - Return immediately after starting mode transition. 1 - Wait until mode transition is complete before returning. This can be useful when setting the mode to a nonzero value. When setting the mode to zero, it is typically not necessary to specify *wait* to 1.

Result set

ttDbWriteConcurrencyModeSet returns no result set:

Examples

The following example shows how to enable standard optimization techniques and return immediately after starting the operation:

```
Command> CALL ttDbWriteConcurrencyModeSet(0,0);
```

Notes

When the mode is set to one (1), all transaction and statement table lock hints are suppressed. This affects hint-triggered `Sn` table locks for `SELECT` statements and subqueries and also hint-triggered `W` table locks for DML statements. Suppression of the table lock hint also suppresses other table-lock hint driven execution plans such as star joins. Regardless of the mode setting, table locks that are not triggered by table-lock hints are not affected.

See also

`ttDbWriteConcurrencyModeGet`

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.

Calling `ttDurableCommit` also makes durable the current transaction and any previously committed delayed durability transactions. There is no effect on other transactions that are committed after 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.

Examples

```
CALL ttDurableCommit;
```

Notes

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 (UPS).

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.

The `ttGridAttach` built-in procedure automatically starts the cache agent if it is not already running. In addition, the `ttGridAttach` built-in procedure sets the specified TCP/IP port for the cache agent to facilitate global cache groups.

To retrieve the information set by this procedure call the built-in procedure ttGridNodeStatus.

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_VARCAR (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_VARCAR (30)`	Fully qualified name that uniquely identifies the grid member for the standby master database.
IPAddr2	`TT_VARCHAR (128)`	IP address of the node where the standby master database resides.

Parameter	Type	Description
port2	TT_INTEGER	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);
```

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
ttGridNameSet
ttGridNodeStatus
"Configuring a cache grid" in *Oracle TimesTen Application-Tier 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 parameters:

Parameter	Type	Description
cvName	`TT_VARCHAR (30)`	The name of the cache group to be checked. If `NULL`, all cache groups owned by the owner, if one is specified, are checked. If `NULL`, and no owner is specified, 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.

Examples

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`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle TimesTen Application-Tier Database Cache User's Guide*

ttGridCreate

Description

This procedure creates a cache grid. Run this procedure only one time to create a grid. You can run it from any standalone database or from the active or standby master database in an active standby pair.

You must commit after calling this procedure if `AUTOCOMMIT=0`.

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.

Examples

To create a grid named `mygrid`:

```
CALL ttGridCreate ('mygrid');
```

See also

`ttGridAttach`
`ttGridCheckOwner`
`ttGridDestroy`
`ttGridDetach`
`ttGridDetachAll`
`ttGridDetachList`
`ttGridGlobalCGResume`
`ttGridGlobalCGSuspend`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle TimesTen Application-Tier 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 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 cache groups. Valid value is `1`.

Result set

`ttGridDestroy` returns no results.

Examples

To destroy the `mygrid` cache grid with force, use:

```
CALL ttGridDestroy ('mygrid', 1);
```

See also

ttGridAttach
ttGridCheckOwner
ttGridCreate
ttGridDetach
ttGridDetachAll
ttGridDetachList
ttGridGlobalCGResume
ttGridGlobalCGSuspend
ttGridNameSet
ttGridNodeStatus

"Configuring a cache grid" in *Oracle TimesTen Application-Tier 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] [oraclePropWaitSec])
```

Parameters

`ttGridDetach` has the optional parameters:

Parameter	Type	Description
nodeMemberName	`TT_VARCHAR (200)`	Specifies the node to detach from the grid. Each node of an active standby pair must be detached separately.
force	`TT_INTEGER`	Forces a node to be detached without checking whether it is terminated. Valid value is `1`.
oraclePropWaitSec	`TT_INTEGER`	Specifies the number of seconds to wait for all transactions to propagate to the Oracle database before detaching the node. A value of `-1` indicates to wait forever. If no value is specified, `ttGridDetach` waits 1 second.

Result set

`ttGridDetach` returns no results.

Examples

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

ttGridNameSet
ttGridNodeStatus
"Configuring a cache grid" in *Oracle TimesTen Application-Tier 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([oraclePropWaitSec])
```

Parameters

`ttGridDetachAll` has the optional parameter:

Parameter	Type	Description
oraclePropWaitSec	`TT_INTEGER`	Specifies the number of seconds to wait for all transactions to propagate to the Oracle database before detaching all nodes. A value of `-1` indicates to wait forever. If no value is specified, `ttGridDetachAll` waits 1 second.

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`
`ttGridNameSet`
`ttGridNodeStatus`
"Configuring a cache grid" in *Oracle TimesTen Application-Tier 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] [oraclePropWaitSec])
```

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.
oraclePropWaitSec	TT_INTEGER	The optional parameter specifies the number of seconds to wait for all transactions to propagate to the Oracle database before detaching the listed nodes. A value of -1 indicates to wait forever. If no value is specified, ttGridDetachList waits 1 second.

Result set

ttGridDetachList returns no results.

Examples

```
CALL ttGridDetachList('TTGRID_cacheact_3A TTGRID_cachestand_3B',1, );
```

See also

ttGridAttach
ttGridCheckOwner
ttGridCreate
ttGridDetach
ttGridDetachAll
ttGridDestroy
ttGridGlobalCGResume
ttGridGlobalCGSuspend
ttGridNameSet
ttGridNodeStatus
"Configuring a cache grid" in *Oracle TimesTen Application-Tier Database Cache User's Guide*

ttGridFirstMemberAttach

Description

This procedure is similar to the ttGridAttach built-in procedure. Call this procedure instead to attach the first member of the grid in the case that all attached members of the grid have died.

Call this procedure only from one member. Ensure that the cache agents of all other members are stopped before calling this built-in procedure or the procedure fails.

The ttGridFirstMemberAttach built-in procedure automatically starts the cache agent if it is not already running. In addition, the ttGridFirstMemberAttach built-in procedure sets the specified TCP/IP port for the cache agent to facilitate global cache groups.

To retrieve the information set by this procedure call the built-in procedure ttGridNodeStatus.

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:

```
ttGridFirstMemberAttach(currentNode, 'name1', IPAddr1, port1)
```

For a node of an active standby pair:

```
ttGridFirstMemberAttach(currentNode, 'name1',
    IPAddr1, port1 'name2', IPAddr2, port2)
```

Parameters

ttGridFirstMemberAttach 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_VARCHAR (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.

Parameter	Type	Description
name2	TT_VARCAR (30)	Fully qualified name that uniquely identifies the grid member for the standby master database.
IPAddr2	TT_VARCHAR (128)	IP address of the node where the standby master database resides.
port2	TT_INTEGER	Port number for the cache agent process of the standby master database.

Result set

ttGridFirstMemberAttach returns no results.

Examples

To attach to a standalone TimesTen database to a grid:

```
CALL ttGridFirstMemberAttach (1, 'alone2','sys2',5002);
```

See also

ttGridCheckOwner
ttGridCreate
ttGridDestroy
ttGridDetach
ttGridDetachList
ttGridDetachAll
ttGridGlobalCGResume
ttGridGlobalCGSuspend
ttGridNameSet
ttGridNodeStatus
"Configuring a cache grid" in *Oracle TimesTen Application-Tier Database Cache User's Guide*

ttGridGlobalCGResume

Description

This procedure resumes operations that were blocked after a call to ttGridGlobalCGSuspend.

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
ttGridNameSet
ttGridNodeStatus
"Configuring a cache grid" in *Oracle TimesTen Application-Tier Database Cache User's Guide*

ttGridGlobalCGSuspend

Description

This procedure temporarily blocks dynamic loading and deleting cache instances for global cache groups. 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	The number of seconds that the command waits for a pending delete to be propagated to the Oracle database 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
ttGridNameSet
ttGridNodeStatus
"Configuring a cache grid" in *Oracle TimesTen Application-Tier 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, . . .>
major1	TT_VARCHAR (10)	The first number of the major TimesTen release associated with the grid. For example, 11, if the release is 11.2.2.
major2	TT_VARCHAR (10)	The second number of the major TimesTen release associated with the grid. For example, 2, if the release is 11.2.2.
major3	TT_VARCHAR (10)	The third number of the major TimesTen release associated with the grid. For example, 2, if the release is 11.2.2.

Examples

To get information on the mygrid cache grid, use:

```
CALL ttGridInfo ('mygrid');
< MYGRID, CACHEUSER, Linux Intel x86, 32-bit, 11, 2, 2 >
```

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 TimesTen Application-Tier 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.

Examples

To associate the database with the grid `mygrid`, use:

```
CALL ttGridNameSet('mygrid');
```

See also

ttGridAttach
ttGridCheckOwner
ttGridCreate
ttGridDestroy
ttGridDetach
ttGridDetachAll
ttGridDetachList
ttGridGlobalCGResume
ttGridGlobalCGSuspend
ttGridNodeStatus
"Configuring a cache grid" in *Oracle TimesTen Application-Tier 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.
nodeID	TT_INTEGER NOT NULL	The unique ID of the grid node.
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.
memberName1	TT_VARCHAR (200) NOT NULL	The unique member name for the standalone database or active standby database.
IPaddr1	TT_VARCHAR (128) NOT NULL	The IP address where the active master or standalone database is located.
port1	TT_INTEGER NOT NULL	The port number for the cache agent process of the active master or standalone database.
node2Attached	CHAR (1)	Indicates if the standby node is attached to the grid: T - The standby is attached. F - The standby is detached.

Column	Type	Description
host2	TT_VARCHAR (200)	The host name where the standby master database is located.
memberName2	TT_VARCHAR (200)	The unique member name for the standalone 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.

Examples

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
ttGridNameSet
"Configuring a cache grid" in *Oracle TimesTen Application-Tier 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.

Use this procedure 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 host name is returned.

Examples

```
CALL ttHostNameGet ();
```

See also

`ttHostNameSet`
"Setting Up a Replicated System" in *Oracle TimesTen In-Memory Database 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 enables an application to set a default host name for the current session that Replication DDL processing uses 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. To clear the default value, specify `NULL`.

Result set

`ttHostNameSet` returns no results.

Examples

```
CALL ttHostNameSet ('alias1');
```

Notes

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 Replication Guide*

ttIndexAdviceCaptureDrop

Description

This procedure drops existing capture data for either the current connection or for the database. Subsequent calls to `ttIndexAdviceCaptureOutput` at that level return no rows.

This procedure and the procedures related to it are referred to as the Index Advisor. For details on using these procedures, see "Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires no privileges to drop a connection level capture.

This procedure requires `ADMIN` privileges to drop a database level capture.

Syntax

```
ttIndexAdviceCaptureDrop([captureLevel])
```

Parameters

`ttIndexAdviceCaptureDrop` has this optional parameter:

Parameter	Type	Description
captureLevel	`TT_INTEGER`	Legal values for the capture level are: `0` - Index advice capture is dropped at the connection level for the current connection. This is the default. `1` - Index advice capture is dropped at the database level.

Result set

`ttIndexAdviceCaptureDrop` returns no results.

Examples

```
CALL ttIndexAdviceCaptureDrop;
```

Notes

To drop both connection level and database level captures, invoke the command twice, once for each capture level.

It is an error to call this command while a capture is in progress at the level you are attempting to drop.

See also

`ttIndexAdviceCaptureEnd`
`ttIndexAdviceCaptureInfoGet`
`ttIndexAdviceCaptureOutput`
`ttIndexAdviceCaptureStart`
"Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*

ttIndexAdviceCaptureEnd

Description

This procedure ends either an active connection level capture from the current connection or an active database level capture.

This procedure and the procedures related to it are referred to as the Index Advisor. For details on using these procedures, see "Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires no privilege to end a connection level capture.

This procedure requires `ADMIN` privileges to end a database level capture.

Syntax

```
ttIndexAdviceCaptureEnd([captureLevel])
```

Parameters

`ttIndexAdviceCaptureEnd` has this optional parameter:

Parameter	Type	Description
captureLevel	`TT_INTEGER`	Legal values for the capture level are: `0` - Ends index advice capture at the connection level for the current connection. This is the default. `1` - Ends index advice capture at the database level.

Result set

`ttIndexAdviceCaptureEnd` returns no results.

Examples

The following example ends the collection for the connection level capture:

```
Call ttIndexAdviceCaptureEnd(0)
```

Notes

To end both connection level and database level captures, invoke the command twice, once for each capture level.

It is an error to call this procedure without first starting a capture at the specified level by calling the `ttIndexAdviceCaptureStart` procedure.

See also

`ttIndexAdviceCaptureDrop`
`ttIndexAdviceCaptureInfoGet`
`ttIndexAdviceCaptureOutput`
`ttIndexAdviceCaptureStart`
"Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*

ttIndexAdviceCaptureInfoGet

Description

This procedure returns a row for each active capture. A capture is active if it has started capturing index advice or if it has stopped capturing index advice, but the capture data is still available.

One row relates to a connection level capture, if one exists. Another row relates to a database level capture, if one exists. At most there is one connection level and one database capture.

If no capture is in progress or no data exists, this procedure does not return any rows.

This procedure and the procedures related to it are referred to as the Index Advisor. For details on using these procedures, see "Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires no privilege to get information on a connection level capture.

This procedure requires `ADMIN` privileges to get information on a database level capture.

Syntax

```
ttIndexAdviceCaptureInfoGet()
```

Parameters

`ttIndexAdviceCaptureInfoGet` has no parameters.

Result set

`ttIndexAdviceCaptureInfoGet` returns the result set:

Columns	Type	Description
captureState	`TT_INTEGER NOT NULL`	The state of the capture: `0` - A capture is not in progress. `1` - A capture is in progress.
connID	`TT_INTEGER`	The connection ID of the connection that initiated the last capture, or the current capture if one is in progress. This row is not returned if no capture has been initiated.
captureLevel	`TT_INTEGER`	The level of the most recent capture. This row is not returned if no capture has been initiated.
captureMode	`TT_INTEGER`	The mode of the most recent capture. This row is not returned if no capture has been initiated.

Columns	Type	Description
numPrepared	TT_INTEGER	The number of prepared statements during the capture period. This value is NULL if no capture has been initiated.
numExecuted	TT_INTEGER	The number of executed statements during the capture period. This value is NULL if no capture has been initiated.
captureStartTime	TT_TIMESTAMP	The time stamp taken at the start of the capture period. This row is not returned if no capture has been initiated.
captureEndTime	TT_TIMESTAMP	The time stamp taken at the end of the capture period. This value is NULL if no capture is still in progress.

Examples

This example shows capture information for a completed connection level capture for 363 prepared statements and 369 executed statements:

```
Command> CALL ttIndexAdviceCaptureInfoGet();
< 0, 1, 0, 0, 363, 369, 2012-07-27 11:44:08.136833,
2012-07-27 12:07:35.410993 >
1 row found.
```

Notes

If there is an active database level capture and you call this procedure on a connection that does not have ADMIN privilege, TimesTen returns an error.

See also

ttIndexAdviceCaptureDrop
ttIndexAdviceCaptureEnd
ttIndexAdviceCaptureOutput
ttIndexAdviceCaptureStart
"Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*

ttIndexAdviceCaptureOutput

Description

This built-in returns a list of index recommendations from the last recorded capture at the specified level. It also returns an executable `CREATE INDEX` SQL statement for creating the recommended index.

This procedure and the procedures related to it are referred to as the Index Advisor. For details on using these procedures, see "Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*.

For a connection level capture, run this procedure in the same connection that initiated the capture. For a database level capture, run this procedure in a connection with `ADMIN` privileges.

Required privilege

This procedure requires no privilege to get output on a connection level capture.

This procedure requires `ADMIN` privileges to get output on a database level capture.

Syntax

```
ttIndexAdviceCaptureOutput([captureLevel])
```

Parameters

`ttIndexAdviceCaptureOutput` has this optional parameter:

Parameter	Type	Description
captureLevel	`TT_INTEGER`	Legal values for the capture level are: `0` - Outputs index advice at the connection level for the current connection. This is the default value. `1` - Outputs index advice at the database level.

Result set

`ttIndexAdviceCaptureOutput` returns the result set:

Column	Type	Description
`stmtCount`	`TT_INTEGER`	The number of statements in the captured workload that would have benefited from this index if it were present.
createStmt	`TT_VARCHAR (8300) NOT NULL`	The executable statement that can create the recommended index.

Examples

The following example provides the `CREATE INDEX` statement for an index called `PURCHASE_i1` on the `HR.PURCHASE` table. There are four distinct statements that would benefit from the index in this SQL workload.

```
CALL ttIndexAdviceCaptureOutput();
< 4, create index PURCHASE_i1 on HR.PURCHASE(AMOUNT); >
1 row found.
```

Notes

All names returned are fully schema qualified.

See also

ttIndexAdviceCaptureDrop
ttIndexAdviceCaptureEnd
ttIndexAdviceCaptureInfoGet
ttIndexAdviceCaptureStart
"Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*

ttIndexAdviceCaptureStart

Description

This procedure enables index advice capture. It is recommended that statistics be updated before you call this procedure, using `ttOptEstimateStats` and setting the `'invalidate'` parameter set to `'yes'`. Updating the statistics in this way ensures statistics are up to date and forces statements to be re-prepared during the capture. To set statistics to known values instead, call `ttOptSetTblStats` with the `'invalidate'` parameter set to `'yes'`.

This procedure and the procedures related to it are referred to as the Index Advisor. For details on using these procedures, see "Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires no privilege to start a connection level capture.

This procedure requires `ADMIN` privileges to start a database level capture.

Syntax

```
ttIndexAdviceCaptureStart([captureLevel], [captureMode])
```

Parameters

`ttIndexAdviceCaptureStart` has these optional parameters:

Parameter	Type	Description
captureLevel	`TT_INTEGER`	Legal values for the capture level are: `0` - Outputs index advice at the connection level for the current connection. This is the default value. `1` - Outputs index advice at the database level.
`captureMode`	`TT_INTEGER`	Legal values for the capture mode are: `0` - Provides complete capture of index advice including execution of the SQL statements. This is the default. `31` - Capture is based only on the computed statistics and plan analysis. Queries (`SELECT` statements only) are prepared but not executed. This mode can only be used with connection level captures (`captureLevel=0`).

Result set

`ttIndexAdviceCaptureStart` returns no results

Examples

The following example starts a collection for the Index Advisor at the connection-level.

```
Call ttIndexAdviceCaptureStart(0,0);
```

Notes

It is an error to call this procedure if index advice is already being captured at the level specified by the *captureLevel* parameter or at the connection level if no level is specified. Connection level captures can be issued concurrently on independent connections without conflict. Outstanding connection level captures that are in progress when a database level capture begins complete as intended.

See also

ttIndexAdviceCaptureDrop
ttIndexAdviceCaptureEnd
ttIndexAdviceCaptureInfoGet
ttIndexAdviceCaptureOutput
"Using the Index Advisor to recommend indexes" in the *Oracle TimesTen In-Memory Database Operations Guide*

ttLoadFromOracle

Description

This procedure takes a TimesTen table name, an Oracle `SELECT` statement and the number of threads for parallel load. It executes the query on the Oracle database and loads the result set into the specified TimesTen table.

No character set conversion is performed when loading data from an Oracle database into a TimesTen table. The TimesTen database and the Oracle database must use the same character set.

The procedure requires the connection attribute `UID`, the connection attribute `OraclePWD` and the connection attribute `OracleNetServiceName` to be specified. You must commit after calling this procedure.

For more details and usage information, see "Loading data from an Oracle database into a TimesTen table" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires `INSERT` privileges to the table to be loaded.

The session must have all the required privileges to execute the query on the Oracle database.

Syntax

```
ttLoadFromOracle(['tblOwner'], 'tblName', 'Query' [,numThreads])
```

Parameters

`ttLoadFromOracle` has these parameters:

Parameter	Type	Description
tblOwner	`TT_CHAR (30)`	TimesTen table owner (optional). If not provided, the connection ID is used.
tblName	`TT_CHAR (30) NOT NULL`	Name of the table to be loaded with data from the Oracle database. You can use the built-in procedure `ttTableSchemaFromOraQueryGet` to get a schema with which to build the table, if one does not already exist. The specified TimesTen table cannot be a system table, a synonym, a view, a materialized view or a detail table of a materialized view, a global temporary table or a cache group table.
Query	`TT_VARCHAR (409600) NOT NULL`	A `SELECT` query on an Oracle database to derive the table column definition. The query on an Oracle database cannot have any parameter bindings. Provide any expressions in the `SELECT` list with a column alias. Otherwise, an implementation dependent column name is assumed and the expression is not evaluated.

Parameter	Type	Description
numThreads	TT_INTEGER	Number of threads for parallel load (optional). If NULL, defaults to four. Provides parallel loading for tables. Specifies the number of loading threads to run concurrently. One thread performs the bulk fetch from Oracle and the other threads perform the inserts into TimesTen. Each thread uses its own connection or transaction. The minimum value for NumThreads is 2. The maximum value is 10. If you specify a value greater than 10, TimesTen assigns the value 10.

Result set

ttLoadFromOracle returns the result set:

Column	Type	Description
numRows	TT_BIGINT NOT NULL	A single number indicating the number of rows loaded.

Examples

The following example selects information about employees from the Oracle database HR.EMPLOYEES table and loads it into the TimesTen HR.EMPLOYEES table. In this example information was found for 107 employees.

```
Command> CALL ttLoadFromOracle ('HR','EMPLOYEES',
'SELECT * FROM HR.EMPLOYEES');
< 107 >
1 row found.
```

Notes

TimesTen does not empty the table before the load.

The target table does not require a primary key.

TimesTen returns an error if the query output cannot be converted to rows in the target table due to a mismatch of column types or number of columns.

Loading data into TimesTen LOB columns is not supported. If the query on the Oracle database has LOB output, it is mapped to a VAR type.

The load process does not check that the column data types and sizes in the TimesTen table match the data types and sizes of the result set. Instead, the insert is attempted and if the column data types cannot be mapped or the Oracle Database data from the SQL query exceeds the TimesTen column size, TimesTen returns an error.

LOB columns are truncated to 4 MB.

When a table is altered to add columns, secondary partitions are added. Loading a table with multiple partitions is not supported by ttLoadFromOracle.

See also

ttTableSchemaFromOraQueryGet

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` connection attribute. See "LockLevel" on page 1-63 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.

Examples

```
CALL ttLockLevel ('Row');
```

Notes

This procedure does not affect the current transaction.

Row-level locking is required when caching tables from an Oracle database.

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 enables 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. Valid values are 0.0 to 1000000.0 inclusive.

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 error `TT6003` to the application. By default, TimesTen uses a value

of 10 seconds for lock timeouts. If a value of 0 is specified, transactions do not wait for any unavailable locks.

See also

ttLockLevel
"LockLevel" on page 1-63

ttLogHolds

Description

This procedure returns information about transaction log holds, including those created on behalf of incremental backups, replication peers, active standby pairs (and any subscribers), AWT cache groups, 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.

Applications should monitor log holds and the accumulation of log files. For more information, see "Show replicated log holds" in the *Oracle TimesTen In-Memory Database Replication Guide* and "Monitoring accumulation of transaction log files" in the *Oracle TimesTen In-Memory Database Operations Guide*.

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_BIGINT 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 standby master ■ The name of the replication subscriber ■ _ORACLE when tracking AWT cache group propagation ■ 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 XID (transaction ID) of the XA transaction ■ The TimesTen transaction ID of the long-running transaction

Examples

```
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 >
```

The following example shows the output of ttLogHolds built-in procedure for an active standby pair replication scheme, where the active master is master1 and the standby master is master2 with a single subscriber, subscriber1.

```
Command> call ttLogHolds;
< 0, 3569664, Checkpoint                    , master1.ds0 >
< 0, 15742976, Checkpoint                    , master1.ds1 >
< 0, 16351496, Replication                   , ADC6160529:SUBSCRIBER1 >
< 0, 16351640, Replication                   , ADC6160529:MASTER2 >
4 rows found.
```

The following example shows the progress of the asynchronous propagation for an AWT cache group to the Oracle database. The description field contains "_ORACLE" to identify the transaction log hold for the AWT cache group propagation.

```
Command> call ttLogHolds();
< 0, 18958336, Checkpoint                    , cachealone1.ds0 >
< 0, 19048448, Checkpoint                    , cachealone1.ds1 >
< 0, 19050904, Replication                   , ADC6160529:_ORACLE >
3 rows found.
```

ttMonitorHighWaterReset

Description

This procedures sets the value of PERM_IN_USE_HIGH_WATER column in the MONITOR table to the current value of the PERM_IN_USE_SIZE column 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 column. 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.

Examples

```
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 Views 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.

Examples

```
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

ttOptCmdCacheInvalidate

Description

This built-in procedure either forces a recompilation should a dependent command be invoked again, or removes such command from the cache and it must be re-prepared by the user.

Scenarios in which you may want to call this procedure include:

- After all needed statistics have been collected.
- When table cardinalities have been changed significantly.

The procedure either marks a command as needing recompilation or as invalidated.

Neither option stops execution of a command.

Required privilege

This procedure requires the `DDL` privilege.

Syntax

```
ttOptCmdCacheInvalidate('tblName', invalidate)
```

Parameters

`ttOptCmdCacheInvalidate` has these parameters:

Parameter	Type	Description
tblname	`TT_CHAR(61)`	The name of the table for which the dependent commands should be invalidated or recompiled.
invalidate	`TT_INTEGER`	Forces recompilation or invalidates the dependent commands. `1` - Indicates that the commands should be recompiled. The command is recompiled during its first use after calling this built-in procedure. (default) `2` - Indicates that the commands should be invalidated. The command is not reused or recompiled again. If you call the command after you have marked it for invalidation, TimesTen returns an error.

Result set

`ttOptCmdCacheInvalidate` returns no results.

Examples

To recompile dependent commands on the table tab1, use:

```
CALL ttOptCmdCacheInvalidate ('tab1', 1);
```

To invalidate the dependent commands on table tab1, use:.

```
CALL ttOptCmdCacheInvalidate ('tab1', 2);
```

See also

ttOptClearStats
ttOptEstimateStats
ttOptSetColIntvlStats
ttOptSetFlag
ttOptSetOrder
ttOptSetTblStats
ttOptUpdateStats
ttPLSQLMemoryStats

ttOptEstimateStats

Description

The `ttOptEstimateStats` 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`').

The procedure operates on all tables of the current user if *`tblName`* is not specified.

Required privilege

This procedure requires no privilege if the user is the table owner, or if *`tblName`* is not specified.

This procedure requires the `ALTER ANY TABLE` privilege if the 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 an 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 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, `ttOptEstimateStats` does not hold a lock on tables or rows when computing statistics. Computing statistics can still slow performance. Estimating statistics generally provides better performance than computing exact statistics.

If you estimate or update statistics with an empty table list, statistics on system tables are updated also, if you have privileges to update the system tables.

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.get`*`XXX`*`()` 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, 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 2-137 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.

Examples

```
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.get`*`XXX`*`()` 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 enables 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 is 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.
		Because this parameter is a compound structure it cannot be parameterized using ODBC functions or described using the `ttIsql describe` command. For example, a statement like the following fails: `SQLPrepare(hstmt, "call ttOptSetColIntvlStats('t1', 'c1', 1, ?)", SQL_NTS))`.

Result set

`ttOptSetColIntvlStats` returns no results.

Examples

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 enables 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.
num_Uniq	`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.
num_Null	`TT_INTEGER`	Indicates the total number of `NULL`s in the column.

Result set

`ttOptSetColStats` returns no results.

Examples

```
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 enables 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.

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
`BranchAndBound`	Enables or disables branch and bound optimization. If enabled, TimesTen calculates the maximum cost of the query plan during a "zero phase," at the very beginning of the optimization process. If disabled, TimesTen does not perform this cost analysis.
`DynamicLoadEnable`	Enables or disables dynamic load of data from an Oracle database to a TimesTen dynamic cache group. By default, dynamic load of data from an Oracle database 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.

Flag	Description
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.
ForceCompile	Enables or disables forced compilation. If enabled, TimesTen recompiles the query and regenerates the query plan each time. If disabled, TimesTen does not compile the query plan even if it is available.
GenPlan	Enables or disables the creation of entries in the PLAN table for the rest of the transaction. For an example, see "Instruct TimesTen to store the plan in the system PLAN table" in *Oracle TimesTen In-Memory Database Operations Guide*.
GlobalLocalJoin	Enables or disables global execution of SELECT and UNLOAD CACHE GROUP statements that contain a join in a grid. By default, these statements are executed locally. 0 - SELECT and UNLOAD CACHE GROUP statements and the join both are executed locally. 1 - SELECT and UNLOAD CACHE GROUP statements are executed globally, the join is executed locally. See *Oracle TimesTen Application-Tier Database Cache User's Guide* for more details.
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 TimesTen Application-Tier 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.
IndexedOR	Enables or disables serialized table scans. If disabled, TimesTen uses serialized table scans for IN. . . list conditions, else TimesTen uses multiple index scans for an OR condition.
MergeJoin	Enables or disables the use of merge joins.
NestedLoop	Refers to a common way of joining two tables.
NoRemRowIdOpt	Enables or disables internal generation of RowIDs. If enabled, RowIDs are not internally generated for optimization purposes. If disabled, RowIDs may be internally generated, even if the row is not in the SELECT list.

Flag	Description
PassThrough	Temporarily changes the pass through level for TimesTen Cache applications. The pass through level can be set at any time and takes effect immediately. Legal values for this flag are: 0 - (default) - SQL statements are executed only on TimesTen. 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.
Range	Enables or disables the use of existing range indexes in indexed table scans.
Rowid	Enables or disables the use of Row IDs.
RowLock	Allows or disallows the optimizer to consider using row locks.
Scan	Refers to full table scans.
ShowJoinOrder	Shows the join order of the tables in an optimizer scan.
TblLock	Enables or disables the optimizer to consider using table locks.
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.
TmpRange	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.

Flag	Description
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.
UseBoyerMooreStringSearch	Enables or disables the Boyer-Moore string search algorithm. If enabled, Boyer-Moore string search algorithm is enabled. This can improve performance of LIKE operations.

In addition, you can use the string AllFlags to refer to all optimizer flags, and the string Default 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 ( 'MergeJoin', 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.

TblLock status	RowLock status	Effect on the optimizer
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 or when there are very few concurrent transactions accessing the table.

Result set

`ttOptSetFlag` returns no results.

Examples

```
CALL ttOptSetFlag ('TmpHash', 1);
```

Notes

You can also set the join order using statement level optimizer hints in certain SQL statements. For details, see "Statement level optimizer hints" in the *Oracle TimesTen In-Memory Database SQL Reference*. Specifically, see the table, "Differences between statement level and transaction level optimizer hints" to understand the behavior of each style of hint.

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.

Examples

```
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.

Examples

```
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 table names or table 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. `AutoCommit` must be set to `OFF` when running this built-in procedure.

Required privilege

This procedure requires no privilege.

Syntax

```
ttOptSetOrder('joinOrder')
```

Parameters

`ttOptSetOrder` has the required parameter:

Parameter	Type	Description
join_Order	`TT_VARCHAR(1024)`	List of space-separated table or 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');
```

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');
```

Consider an application that makes the following 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

A table alias name for a derived table is not supported in the join order. If you specify a table alias name, TimesTen returns the warning message 965 that indicates the order cannot be honored.

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, Times issues a warning and ignores the specified join order completely.

You can also set the join order using statement level optimizer hints in certain SQL statements. For details, see "Statement level optimizer hints" in the *Oracle TimesTen In-Memory Database SQL Reference*. Specifically, see the section, "Differences between statement level and transaction level optimizer hints" to understand the behavior of each style of hint.

See also

ttOptEstimateStats
ttOptGetFlag
ttOptGetOrder
ttOptSetColIntvlStats
ttOptSetFlag
ttOptSetTblStats
ttOptUpdateStats

ttPLSQLMemoryStats

ttOptSetTblStats

Description

This procedure modifies the statistics for the specified table. This procedure enables 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.
num_Rows	`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.

Examples

```
CALL ttOptSetTblStats ( 'ACCTS', 10000, 0 );
```

Notes

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 is 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 Views 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 the order of the 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
joinOrder	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.

Examples

```
>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

ttOptStatsExport

Description

This procedure returns the set of statements required to restore the table statistics to the current state. If no table is specified, it returns the set of statements required to restore the table statistics for all user tables that the calling user has permission to access.

Required privilege

This procedure requires `ADMIN` privilege.

Syntax

```
ttOptStatsExport('tblName')
```

Parameters

`ttOptStatsExport` has the parameter:

Parameter	Type	Description
tblName	`TT_CHAR (61) NOT NULL`	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.

Result set

`ttOptStatsExport` returns the result set:

Column	Type	Description
stmt	`TT_VARCHAR (8300) NOT NULL`	The set of statements required to restore the table(s) statistics to the current state.

Examples

```
CALL ttOptStatsExport('MyTable');
```

See also

"Create script to regenerate current table statistics" in the *Oracle TimesTen In-Memory Database Operations Guide*.

ttOptUpdateStats

Description

This procedure updates the statistics for the specified table. TimesTen looks at the data 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.

The procedure operates on all tables of the current user if *tblName* is not specified.

Required privilege

This procedure requires no privilege if the user is the table owner, or if *tblName* is not specified.

This procedure requires the ALTER ANY TABLE privilege if the 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, marks all commands for reprepare on next execution except ALTER TABLE DROP TABLE, and the ALTER TABLE ADD COLUMN FOR SELECT * FROM TABLE statements. These exceptions require manual reprepare. 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. The *option* parameter is optional and defaults to 0. See the notes below for more information.

Result set

ttOptUpdateStats returns no results.

Examples

```
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 values in the column
- Total number of NON NULL UNIQUE values in the column
- Total number of rows in the table
- Interval information, where 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 2-126 for information on estimating statistics.

If you estimate or update statistics with an empty table list, statistics on system tables are updated also, if you have privileges to update the system tables.

See also

ttOptEstimateStats
ttOptGetColStats
ttOptSetColStats
ttOptSetColIntvlStats
ttOptSetTblStats
ttOptUpdateStats

ttOptUseIndex

Description

This procedure enables applications to alter the generation of execution plans by the TimesTen query optimizer. Applications can call this procedure 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.

`AutoCommit` must be set to `OFF` when running this built-in procedure.

Required privilege

This procedure requires no privilege.

Syntax

```
ttOptUseIndex('IndexName, CorrelationName, 0 | 1 [;...]')
```

Parameters

`ttOptUseIndex` has a single comma-delimited string parameter, *`indOption`*, of type `TT_VARCHAR(1024)` with these components:

Component	Description
`IndexName`	The name of the user-defined index or `'_TMPRANGE'` 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');
```

Notes

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( )
```

Parameters

`ttPLSQLMemoryStats` takes no parameters.

Result Set

`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 MB, 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.
```

ttRamPolicyAutoReloadGet

Description

This procedure returns the RAM autoreload policy used to determine if a database is reloaded into RAM after an invalidation. The policy can be either `autoreload` or `noautoreload`.

Required privilege

This procedure requires no privilege.

Syntax

```
ttRamPolicyAutoReloadGet()
```

Result set

`ttRamPolicyAutoReloadGet` returns the results:

Column	Type	Description
flag	`TT_INTEGER`	The policy used to determine if the database is reloaded into RAM after an invalidation. Valid values are: `0` - The database is not automatically reloaded into memory after an invalidation. This is the equivalent of the command `ttAdmin -noAutoReload`. `1` - The database is automatically reloaded into memory after an invalidation. This is the equivalent of the command `ttAdmin -autoReload`. This is the default autoreload policy.

Parameters

`ttRamPolicyAutoReloadGet` has no parameters.

Examples

To view the RAM autoreload policy, use:

```
CALL ttRamPolicyAutoReloadGet();
```

See also

`ttRamPolicyAutoReloadSet`
"ttAdmin" on page 3-3

ttRamPolicyAutoReloadSet

Description

This procedure determines the RAM autoreload policy if a database is invalidated. The policy can be either `autoreload` or `noautoreload`.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttRamPolicyAutoReloadSet(flag)
```

Parameters

`ttRamPolicyAutoReloadSet` has the parameters:

Parameter	Type	Description
flag	`TT_INTEGER NOT NULL`	The policy used to determine if the database is reloaded into RAM after an invalidation. Valid values are: `0` - The database is not automatically reloaded into memory after an invalidation. This is the equivalent of the command `ttAdmin -noAutoReload`. `1` - The database is automatically reloaded into memory after an invalidation. This is the equivalent of the command `ttAdmin -autoReload`. This is the default autoreload policy.

Result set

`ttRamPolicyAutoReloadSet` returns no results.

Examples

To automatically reload a database into RAM after an invalidation, use:

```
CALL ttRamPolicyAutoReloadSet(1);
```

See also

`ttRamPolicyAutoReloadGet`
"ttAdmin" on page 3-3

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 (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

`ttRamPolicySet`
"ttAdmin" on page 3-3
"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 (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

`ttRamPolicyGet`

"ttAdmin" on page 3-3
"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.

Examples

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.

Examples

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(['subscriber'], ['hostname'])
```

Parameters

`ttReplicationStatus` has the optional parameters:

Parameter	Type	Description
`subscriber`	`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 host name 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`	Name of the system that hosts the subscriber.
`port`	`TT_INTEGER NOT NULL`	TCP/IP port used by the subscriber agent to receive updates from the master. A value of 0 indicates replication has automatically assigned the port.

Column	Type	Description
pState	TT_CHAR(10) NOT NULL	Current replication state of the subscriber with respect to its master database. The values of the result column are: start - Replication is enabled to this peer. pause - Replication is temporarily paused to this peer. TimesTen preserves updates. See "Setting the replication state of subscribers" in *Oracle TimesTen In-Memory Database Replication Guide* for more information. 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 the master database is retaining for a subscriber.
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.

Examples

```
Command> call ttReplicationStatus();
< MASTER2, HOST1, 0, start     , 1, 257142, \
     _ACTIVESTANDBY         , TTREP       >
1 row found.

Command> call ttReplicationStatus('master2', 'host1');
< MASTER2, HOST1, 0, start     , 1, 266439, \
     _ACTIVESTANDBY               , TTREP                          >
1 row found.
```

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 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:

Column	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 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 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 the following.

```
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();
```

Notes

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 - Start (default). Starts replication to the subscriber. 1 - Pause. Pauses the replication agent, preserving updates. 2 - Stop. Stops replication to the subscriber, discarding updates. All other state codes are disallowed. (This procedure cannot set a subscriber state to "failed.") "Setting the replication state of subscribers" in the *Oracle TimesTen In-Memory Database Replication Guide* for more information.

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*

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 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*.
subscriberHostName	TT_VARCHAR(200)	The subscriber's host. May be NULL to indicate all hosts of subscribing peers.

Parameter	Type	Description
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.

Examples

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)` descr	`0` (default) - Don't wait for return notification configured with the `RETURN RECEIPT BY REQUEST` or `RETURN TWOSAFE BY REQUEST` option. `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.
localAction	`TT_INTEGER`	The current `LOCAL ACTION` configuration for `RETURN` services. `1` (default) - `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`. `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.

Examples

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`*.

Applications can call `ttRepSyncGet` 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 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`. 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.
localAction	`TT_INTEGER`	Action to be performed in the event the subscriber cannot 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` (default) - `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`. `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 )
```

Notes

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('subscriber', '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 host name are NULL, then all receiving stores are selected.

Result set

ttRepSyncSubscriberStatus returns:

Column	Type	Description
disabled	TT_INTEGER	Value is either: 1 - The return service has been disabled on the subscriber database. 0 - The return service is still enabled on the subscriber database.

Notes

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 enables 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` (default) - Updates are being transmitted to subscribers on the connection.

Examples

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. Use this procedure 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. (This is the default.) When set to 0, updates are not transmitted to any subscribers for the remainder of the transaction in which this call was issued on the connection that issued it.

Result set

`ttRepTransmitSet` returns no results.

Examples

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(xactID)
```

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 TABLE` or `ALTER TABLE` operation. Setting a user-assigned value for the column ID enables 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.

Examples

```
CALL ttSetUserColumnID('APP.SESSION', 'SESSIONID', 15);
```

See also

`ttSetUserTableID`
Oracle TimesTen In-Memory Database 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.

Examples

```
CALL ttSetUserTableID('APP.SESSION', 0x123456);
```

See also

`ttSetUserColumnID`
Oracle TimesTen In-Memory Database Replication Guide

ttSize

Description

This procedure estimates the size of a table or view and the size of indexes. 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 estimates 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 following 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();

. . . . . .
```

Notes

The ttSize procedure enables you to estimate how large a table will be with its full population of rows based on a small sample. For the best results, populate the table with at least 1,000 typical rows.

See also

ttComputeTabSizes

ttSQLCmdCacheInfo

Description

This procedure returns information about all prepared SQL statements in the TimesTen SQL command cache.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttSQLCmdCacheInfo([sqlCmdID])
```

Parameters

`ttSQLCmdCacheInfo` has the optional parameter:

Parameter	Type	Description
sqlCmdID	`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, information is displayed for all commands.

Result set

`ttSQLCmdCacheInfo` returns the result set:

Column	Type	Description
sqlCmdID	`TT_INTEGER NOT NULL` for 32-bit systems `TT_BIGINT NOT NULL` for 64-bit systems	The unique identifier of a command.
privateCommandConnectionID	`TT_INTEGER`	If the command is private, this is the connection ID of the connection where it was prepared. If not a private command, this value is `2048`.
executions	`TT_BIGINT NOT NULL`	Counts the number of executions of the command.
prepares	`TT_BIGINT NOT NULL`	Counts the number of prepares for the command.
reprepares	`TT_BIGINT NOT NULL`	Counts the number of reprepares for the command.
freeable	`TT_TINYINT NOT NULL`	Indicates whether this command can be garbage collected by the subdaemon. `1` - Indicates freeable. `0` - Indicates non-freeable.

Column	Type	Description
size	TT_INTEGER NOT NULL	The total space (bytes) allocated for this command in the command cache.
owner	TT_CHAR(31) NOT NULL	The user who created the command.
queryText	TT_VARCHAR (409600) NOT NULL	The full SQL text for the current command.

Examples

To display command information in `ttIsql` for all the current valid commands, use:

```
Command> call ttsqlcmdcacheinfo;
< 51635464, 2048, 12, 12, 0, 1, 3056, SYS                          , delete fr
om sys.idl_sb4$ where obj#=:1 and part=:2 >
< 43437072, 2048, 5, 5, 0, 1, 1960, SYS                           , select obj#
 from sys.objerror$ >
< 51620736, 2048, 4, 4, 0, 1, 2736, SYS                           , delete from
 sys.obj$ where obj# = :1 >
< 51680216, 2048, 1, 1, 0, 1, 3592, BWAF4EVR                      , call ttsqlc
mdcacheinfo(51623232) >
< 51676856, 2048, 2, 2, 0, 0, 3552, BWAF4EVR                      , call ttsqlc
mdcacheinfo >
< 43438936, 2048, 5, 5, 0, 1, 3200, SYS                           , select obj#
 from sys.syn$ where owner=:1 and name=:2 >
< 44066504, 2048, 0, 14, 0, 1, 5640, SYS                           , select nul
l from sys.obj$ where obj#=:1 and type#=:2 and obj# not in (select p_obj# from d
ependency$ where p_obj# = sys.obj$.obj#) >
< 51649488, 2048, 1, 1, 0, 1, 2344, BWAF4EVR                      , create tabl
e tab1 (c1 number primary key not null, c2 number) >
< 51671608, 2048, 1, 1, 0, 1, 4656, BWAF4EVR                      , call ttSQLC
mdCacheInfo2(51635464) >
< 51666232, 2048, 1, 1, 0, 1, 2048, BWAF4EVR                      , call ttSQLC
mdCacheInfoGet >
< 51612064, 2048, 4, 4, 0, 1, 8424, SYS                           , select o.ow
ner#, o.name, o.namespace,    o.obj#, d.d_timestamp, nvl(d.property,0), o.type#,
 d.d_attrs  from sys.dependency$ d, sys.obj$ o  where d.p_obj#=:1  and   (d.p_ti
mestamp=nvl(:2,d.p_timestamp) or d.property=2)  and   o.owner#=nvl(:3,o.owner#)
 and   d.d_obj#=o.obj#  order by o.obj# >
< 43415648, 2048, 4, 4, 0, 1, 4544, BWAF4EVR                      , create acti
ve standby pair sampledb_1122, bwaf4evr_dummy1 subscriber bwaf4evr_dummy2 >
< 43431912, 2048, 5, 5, 0, 1, 4720, SYS                           , select owne
r#,name,namespace,obj#,type#,ctime,mtime,stime,status,flags from sys.obj$ where
obj#=:1 >
< 51657712, 2048, 4, 4, 0, 1, 3552, BWAF4EVR                      , call ttSQLC
mdCacheInfo >
< 51653200, 2048, 1, 1, 0, 1, 1816, BWAF4EVR                      , call ttxlab
ookmarkcreate('mybookmark', 0x01) >
< 43420768, 2048, 1, 1, 0, 1, 2064, BWAF4EVR                      , create tabl
e tab1 (c1 number, c2 number) >
< 44058168, 2048, 14, 14, 0, 1, 7760, SYS                          , select o.
owner#,o.obj#,u.name,o.name,o.namespace from sys.user$ u, sys.obj$ o where u.use
r#=o.owner#   and o.type#=:1   and not exists   (select p_obj# from sys.dependen
cy$ where p_obj# = o.obj#) order by o.obj# for update >
< 49370616, 2048, 1, 1, 0, 0, 4024, SYS                           , select u.us
er#, u.password, u.identification, u.astatus from sys.user$ u where u.name = :na
me and u.type# = 1 >
< 51655376, 2048, 2, 2, 0, 1, 2528, BWAF4EVR                      , select * fr
```

```
om tab1 >
< 51638280, 2048, 4, 4, 0, 1, 2544, SYS                              , delete from
 sys.objauth$ where obj#=:1 >
< 43423200, 2048, 14, 14, 0, 1, 5520, SYS                             , select ow
ner#,name,namespace,obj#,type#,ctime,mtime,stime,status,flags from sys.obj$ wher
e owner#=:1 and name=:2 and namespace=:3 >
< 51668216, 2048, 1, 1, 0, 1, 3592, BWAF4EVR                         , call ttSQLC
mdCacheInfo(51635464) >
< 51661208, 2048, 3, 3, 0, 1, 4640, BWAF4EVR                         , call ttSQLC
mdCacheInfo2 >
< 43428992, 2048, 5, 5, 0, 1, 2800, SYS                              , select sys.
objectSequence.nextval from dual >
< 51629120, 2048, 12, 12, 0, 1, 3040, SYS                             , delete fr
om sys.idl_char$ where obj#=:1 and part=:2 >
< 51641192, 2048, 2, 2, 0, 1, 2112, BWAF4EVR                         , create tabl
e tab1 (c1 number not null, c2 number) >
< 43442488, 2048, 5, 5, 0, 1, 4616, SYS                              , insert into
 sys.obj$(owner#,name,namespace,obj#,type#,ctime,mtime,stime,status,flags) value
s(:1,:2,:3,:4,:5,:6,:7,:8,:9,:10) >
< 51632072, 2048, 12, 12, 0, 1, 3040, SYS                             , delete fr
om sys.idl_ub2$ where obj#=:1 and part=:2 >
< 49375216, 2048, 0, 1, 0, 0, 4232, SYS                              , select 1 fr
om sys.sysauth$ s where (s.grantee# = :userid or s.grantee# = 1) and (s.privileg
e# = :priv or s.privilege# = 67) >
< 51626304, 2048, 12, 12, 0, 1, 3040, SYS                             , delete fr
om sys.idl_ub1$ where obj#=:1 and part=:2 >
< 51645776, 2048, 1, 1, 0, 1, 2344, BWAF4EVR                         , create tabl
e tab1 (c1 number primary key not null, col2 number) >
< 51623232, 2048, 4, 4, 0, 1, 2704, SYS                              , delete from
 sys.source$ where obj#=:1 >
32 rows found.
```

To display command information in `ttIsql` for *`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.
```

To display the information formatted vertically in `ttIsql`, use:

```
Command> vertical call ttSQLCmdCacheInfo;
...
```

To display the information vertically in `ttIsql` for *`sqlCmdID`* `51623232`, use:

```
Command> vertical call ttsqlcmdcacheinfo(51623232);

  SQLCMDID:                       51623232
  PRIVATE_COMMAND_CONNECTION_ID:  2048
  EXECUTIONS:                     4
  PREPARES:                       4
  REPREPARES:                     0
  FREEABLE:                       1
  SIZE:                           2704
  OWNER:                          SYS
  QUERYTEXT:                      delete from sys.source$ where obj#=:1

1 row found.
```

See also

ttSQLCmdCacheInfo2
ttSQLCmdCacheInfoGet

ttSQLCmdCacheInfo2

Description

This procedure returns information about all prepared SQL statements in the TimesTen SQL command cache.

It is similar to `ttSQLCmdCacheInfo`, but returns additional columns, as indicated by the result set documentation.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttSQLCmdCacheInfo2([sqlCmdID])
```

Parameters

`ttSQLCmdCacheInfo2` has the optional parameter:

Parameter	Type	Description
sqlCmdID	`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, information is displayed for all commands.

Result set

`ttSQLCmdCacheInfo2` returns the result set:

Column	Type	Description
sqlCmdID	`TT_INTEGER NOT NULL` for 32-bit systems `TT_BIGINT NOT NULL` for 64-bit systems	The unique identifier of a command.
privateCommandConnectionID	`TT_INTEGER`	If the command is private, this is the connection ID of the connection where it was prepared. If not a private command, this value is `2048`.
executions	`TT_BIGINT NOT NULL`	Counts the number of executions of the command.
prepares	`TT_BIGINT NOT NULL`	Counts the number of prepares for the command.
reprepares	`TT_BIGINT NOT NULL`	Counts the number of reprepares for the command.

Column	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 user who created the command.
queryText	TT_VARCHAR (409600) NOT NULL	The full SQL text for the current command.
fetchCount	TT_BIGINT	The total number of fetch executions done for this statement. The number of fetches depends on TT_PREFETCH_COUNT. The pre-fetch count has a default value of 5 in Read Committed isolation mode and a default of 128 in Serializable mode.
startTime	TT_TIMESTAMP	The time when the statement was last executed. The value is in the form: *YYYY-MM-DD HH:MI:SS.FFF*
maxExecuteTime	NUMBER	The maximum wall clock execute time in seconds for this statement.
lastExecuteTime	NUMBER	Last measured execution time in seconds of the command.
minExecuteTime	NUMBER	If SqlCmdSampleFactor > 0, minimum execute time in seconds, otherwise 0.0.

Examples

To display command information in ttIsql for all the current valid commands, use:

```
Command> call ttSQLCmdCacheInfo2;
...
```

The following example shows the difference in output between ttSQLCmdCacheInfo and ttSQLCmdCacheInfo2:

```
Command> call ttSQLCmdCacheInfo;
...
< 51635464, 2048, 12, 12, 0, 1, 3056, SYS , delete from
sys.idl_sb4$ where obj#=:1 and part=:2 >
...

Command> call ttSQLCmdCacheInfo2;
...
< 51635464, 2048, 12, 12, 0, 1, 3056, SYS, delete
from sys.idl_sb4$ where obj#=:1 and part=:2, 0,
2013-10-28 16:47:09.173000, 0, 0,
0 >
...
```

See also

ttSQLCmdCacheInfo
ttSQLCmdCacheInfoGet

ttSQLCmdCacheInfoGet

Description

This procedure displays information about the commands in the TimesTen SQL command cache.

Required privilege

This procedure requires no privilege.

Syntax

```
ttSQLCmdCacheInfoGet()
```

Parameters

`ttSQLCmdCacheInfoGet` has no parameters.

Result set

`ttSQLCmdCacheInfoGet` returns the result set:

Column	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, in bytes.

Examples

To display the command count, freeable command count, and total space allocated to the command cache, use:

```
Command> call ttSQLCmdCacheInfoGet;
< 5,4,12316 >
1 row found
```

See also

ttSQLCmdCacheInfo
ttSQLCmdCacheInfo2

ttSQLCmdQueryPlan

Description

This procedure returns all detailed runtime 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([sqlCmdID])
```

Parameters

`ttSQLCmdQueryPlan` has the optional parameter:

Parameter	Type	Description
sqlCmdID	`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:

Column	Type	Description
sqlCmdID	`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(409600)`	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.
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.
indexName	`TT_CHAR(31)`	Name of the index used in this step, if any.

Column	Type	Description
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, RowLkRangeScan , T2 , TTUSER , I2 , , >
< 528078576, <NULL>, 2, 7, RowLkRangeScan , T5 , TTUSER , I2 , , >
< 528078576, <NULL>, 3, 6, NestedLoop , , , , , >
< 528078576, <NULL>, 4, 6, RowLkRangeScan , 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, RowLkRangeScan , 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, RowLkRangeScan  , T2
 , TTUSER  , I2 , , NOT(LIKE( tuf ,abc ,NULL )) >
< 528079360, <NULL>, 2, 3, RowLkRangeScan  , 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 */
```

ttSQLExecutionTimeHistogram

Description

The `ttSQLExecutionTimeHistogram` built-in procedure returns a histogram of SQL execution times for either a single SQL command or all SQL commands if command cache sampling is enabled.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

`ttSQLExecutionTimeHistogram(`*`sqlCommandID`*`)`

Parameters

`ttSQLExecutionTimeHistogram` 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 command cache.

Result set

`ttSQLExecutionTimeHistogram` returns the result set:

Column	Type	Description
histogramSamples	`TT_BIGINT`	The number of SQL command execution time operations have been measured since either the database was started or the `ttStatsConfig` built-in procedure was used to reset the statistics.
`totalExecuteTime`	`NUMBER`	The accumulated wall clock execution time when sampling in seconds.
`bucketUpperBound`	`NUMBER`	The upper limit in seconds of execution time.
`count`	`TT_BIGINT`	The number of SQL commands with time less than or equal to `ExecutionTimeLimit` and greater than `ExecutionTimeLimit` from the previous row or 0.

Examples

The following example shows the output for the `ttSQLExecutionTimeHistogram` built-in procedure:

The following example of the `ttSQLExecutionTimeHistogram` built-in procedure shows that a total of 1919 statements executed. The total time for all 1919 statements to execute was 1.090751 seconds. This example shows that SQL statements ran in the following time frames:

- 278 statements executed in a time frame that was less than or equal to .00001562 seconds.

- 1484 statements executed in a time frame that was greater than .00001562 seconds and less than or equal to .000125 seconds.
- 35 statements executed in a time frame that was greater than .000125 seconds and less than or equal to .001 seconds.
- 62 statements executed in a time frame that was greater than .001 seconds and less than or equal to .008 seconds.
- 60 statements executed in a time frame that was greater than .008 seconds and less than or equal to .064 seconds.

```
Command> call ttSQLExecutionTimeHistogram;
< 1919, 1.090751, .00001562, 278 >
< 1919, 1.090751, .000125, 1484 >
< 1919, 1.090751, .001, 35 >
< 1919, 1.090751, .008, 62 >
< 1919, 1.090751, .064, 60 >
< 1919, 1.090751, .512, 0 >
< 1919, 1.090751, 4.096, 0 >
< 1919, 1.090751, 32.768, 0 >
< 1919, 1.090751, 262.144, 0 >
< 1919, 1.090751, 9.999999999E+125, 0 >
10 rows found.
```

See also

ttStatsConfig
ttSQLCmdCacheInfo2

ttStatsConfig

Description

The `ttStatsConfig` built-in procedure controls statistics collection and parameters. This procedure takes a name/value pair as in put and outputs a single row result set corresponding to the name/value pair parameters.

Required privilege

This procedure requires the `ADMIN` privilege.

Syntax

```
ttStatsConfig("param", [value])
```

Parameters

`ttStatsConfig` has the parameters:

Parameter	Type	Description
param	`VARCHAR2(50) NOT NULL`	The unique identifier of a SQL command in the TimesTen command cache.
value	`VARCHAR2(200)`	The value of the specified command. If no value is supplied displays the query plan for all valid commands in the TimesTen cache.

Result set

`ttStatsConfig` returns the result set:

Column	Type	Description
param	`VARCHAR2(50) NOT NULL`	The unique identifier of a SQL command in the TimesTen command cache.
value	`VARCHAR2(200)`	The value of the specified command. If no value was supplied, this is the current value of the command.

Parameter / Value Pairs

These name/value pairs can be returned in the result set:

Name	Value	Description
`SQLCmdSampleFactor`	`0 <= value <= 60000`	The frequency at which a SQL command sample is taken. The default is `0`. A value of `0` indicates that sampling is turned off. A value greater than `0` indicates that a sample is taken at that interval of SQL statements. For example, a value of `10` indicates that for every 10th SQL statement executed, the wall clock time of that execution is captured.

Name	Value	Description
ConnSampleFactor	C,S 0<=C<=Connections 0<=S<=60000	The unique identifier of a SQL command in the TimesTen command cache. If you do not supply a value, TimesTen displays the current value of the command.
SQLCmdHistogramReset	0 or not	The existing SQL execution time statistics are reset if the specified value is nonzero.
StatsLevel	NONE TYPICAL ALL BASIC	Specifies the level of collection for database and operating system statistics. TimesTen collects these statistics for a variety of purposes, including making self-management decisions. Setting the StatsLevel parameter to NONE disables the collection of system statistics. The default setting of TYPICAL ensures collection of all major statistics required for database self-management functionality and provides best overall performance. The default value should be adequate for most environments. When the StatsLevel parameter is set to ALL, additional statistics are added to the set of statistics collected with the TYPICAL setting. The additional statistics are timed operating system statistics and plan execution statistics. Setting the StatsLevel parameter to BASIC disables the collection of many of the important statistics required by many TimesTen features.

Examples

Sample every command:

```
Command> call ttStatsConfig('SqlCmdSampleFactor',1);
< SQLCMDSAMPLEFACTOR, 1 >
1 row found.
```

Check whether sampling:

```
Command> call ttStatsConfig('SqlCmdSampleFactor');
< SQLCMDSAMPLEFACTOR, 1 >
1 row found.
```

Sample every fifth statement on connection 1.

```
Command> call ttStatsConfig('ConnSampleFactor', '1,5');
< CONNSAMPLEFACTOR, 1,5 >
1 row found.
```

Turn off sampling on connection 1.

```
Command> call ttStatsConfig('ConnSampleFactor', '1,0');
< CONNSAMPLEFACTOR, 1,0 >
1 row found.
```

Check data store statistics collection level.

```
Command> call ttstatsconfig('StatsLevel');
```

```
< STATSLEVEL, TYPICAL >
1 row found.
```

Turn off data store statistics collection.

```
Command> call ttstatsconfig('StatsLevel','None');
< STATSLEVEL, NONE >
1 row found.
```

ttTableSchemaFromOraQueryGet

Description

This built-in procedure evaluates a SELECT query on a table in an Oracle database and generates a CREATE TABLE SQL statement that you can choose to execute. The TimesTen CREATE TABLE statement matches the result set column names and types.

This procedure does not create the TimesTen table, it only returns a statement that identifies the table schema.

For more details and usage information, see "Loading data from an Oracle database into a TimesTen table" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires no privileges. The session user must have all required privileges to execute the query on the Oracle database.

Syntax

```
ttTableSchemaFromOraQueryGet(['tblOwner'], 'tblName', 'Query')
```

Parameters

ttTableSchemaFromOraQueryGet has the parameters:

Parameter	Type	Description
tblOwner	TT_CHAR (30)	TimesTen table owner (optional). If not provided, the connection ID is used.
tblName	TT_CHAR (30) NOT NULL	Table name for the CREATE TABLE statement. The specified TimesTen table cannot be a system table, a synonym, a view, a materialized view or a detail table of a materialized view, a global temporary table or a cache group table.
Query	TT_VARCHAR (409600) NOT NULL	A SELECT query on an Oracle database to derive the table column definition. Any expressions in the SELECT list should be provided with a column alias; otherwise, an implementation dependent column name is assumed and the expression is not evaluated.

Result set

ttTableSchemaFromOraQueryGet returns the result set:

Column	Type	Description
createSQL	TT_VARCHAR (409600) NOT NULL	A CREATE TABLE statement that matches the result set of the SELECT query on an Oracle database.

Examples

This example, returns the CREATE TABLE statement to create the TimesTen HR.EMPLOYEES table with all columns found in the Oracle database HR.EMPLOYEES table.

```
Command> call ttTableSchemaFromOraQueryGet('hr','employees',
 'SELECT * FROM hr.employees');
< CREATE TABLE "HR"."EMPLOYEES" (
"EMPLOYEE_ID" number(6,0) NOT NULL,
"FIRST_NAME" varchar2(20 byte),
"LAST_NAME" varchar2(25 byte) NOT NULL,
"EMAIL" varchar2(25 byte) NOT NULL,
"PHONE_NUMBER" varchar2(20 byte),
"HIRE_DATE" date NOT NULL,
"JOB_ID" varchar2(10 byte) NOT NULL,
"SALARY" number(8,2),
"COMMISSION_PCT" number(2,2),
"MANAGER_ID" number(6,0),
"DEPARTMENT_ID" number(4,0)
 ) >
1 row found.
```

Notes

The query on the Oracle database cannot have any parameter bindings.

TimesTen returns an error if the query cannot be described on the Oracle database, for example, if there is a syntax error.

If an output column type does not have a matching type in TimesTen, TimesTen outputs a warning and the following line for the column definition: >>>>*column_name column_type /* reason */*

If the query on the Oracle database outputs types not supported by TimesTen, you can add a CAST clause in the SELECT list to explicitly change the output to a TimesTen supported type. Column aliases can be specified for expressions in the SELECT list.

If the query on the Oracle database has LOB output, it is mapped to a VAR type.

ttVersion

Description

The `ttVersion` utility lists the TimesTen release information, including: number, platform, instance name, instance administrator, instance home directory, daemon home directory, port number and build timestamp. You can specify various levels of output:

- You can specify `ttVersion` with no options to list abbreviated output.
- You can specify the `-m` option to list enhanced output.
- You can specify an attribute to list output only for a specific attribute.

Required privilege

This procedure requires no privilege.

Syntax

```
ttVersion()
```

Parameters

`ttVersion` has no parameters.

Result set

`ttVersion` returns the result set:

Column	Type	Description
`major1`	`TT_INTEGER NOT NULL`	The major release number. Indicates releases with major infrastructure and functionality changes.
`major2`	`TT_INTEGER NOT NULL`	The second major release number. Indicates a version with new functionality changes, but no infrastructure changes.
`minor`	`TT_INTEGER NOT NULL`	The minor release number. Indicates a release that contains all bug fixes since the previous maintenance release.
`patch`	`TT_INTEGER NOT NULL`	Indicates a release with minor bug fixes.
`portpatch`	`TT_INTEGER NOT NULL`	Indicates a release with patch fixes for particular platforms.

Examples

```
CALL ttVersion( );
<11, 2, 2 , 5, 0>
1 row found.
```

In this case, the TimesTen release number is: 11.2.2.5.0.

ttWarnOnLowMemory

Description

This procedure enables 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` connection attributes.

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 warnings for the permanent data partition `0` - Disable warnings for the permanent data partition
temporary	`TT_INTEGER NOT NULL`	`1`- Enable warnings for the permanent data partition `0` - Disable warnings for the permanent data partition

Result set

`ttWarnOnLowMemory` returns no results.

Examples

```
CALL ttWarnOnLowMemory(1, 0);
```

Enables low memory warnings for the permanent data partition only.

Notes

By default, TimesTen does not issue low memory warnings 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 distinguish successive transactions of the same transaction ID.

Examples

```
Command > automcommit 0;
Command > call ttXactIdGet;
<2,11>
1 row found
Command > commit;
Command > call ttXactIdGet
<3, 12>
1 row found
```

Notes

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
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.

Examples

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*.

Notes

You can also create a bookmark when you call `ttXlaPersistOpen` function to initialize an XLA handle. See "Creating or reusing a bookmark" in *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.

Examples

```
Command > call ttXlaBookmarkDelete('mybookmark');
```

Notes

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.

Examples

```
Command > call ttXlaSubscribe ('SALLY.ACCTS', mybookmark);
```

Notes

Alternatively, the `ttXlaTableStatus` function subscribes the current bookmark to updates to the specified table, or determines whether the current bookmark is already monitoring DML records associated with the table. See "Specifying which tables to monitor for updates" in *Oracle TimesTen In-Memory Database C Developer's Guide*

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.

Examples

```
Command > call ttXlaUnsubscribe ('SALLY.ACCTS', mybookmark);
```

Notes

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

This chapter provides reference information for TimesTen utilities, beginning with the following introductory sections:

- 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 nonzero if an error occurs.

> **Note:** The utility name and options listed in this chapter are case-insensitive. They appear 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` connection 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.

Generally, when no `UID` connection 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.

Note: For security reasons, we do not recommend setting a a value for `PWD` on 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, TimesTen 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

Enables 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 database 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.

Syntax

```
ttAdmin {-h | -help | -?}

ttAdmin {-V | -version}

ttAdmin [-ramPolicy always|manual|inUse [-ramGrace secs] ]
[-ramLoad] [-ramUnload]
[-autoreload | -noautoreload]
[-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.

Option	Description
DSN	Specifies an ODBC data source name of the database to be administered.
-h -help -?	Prints a usage message and exits.
-autoreload | -noautoreload	if set to -noautoreload, TimesTen does not automatically reload the database after an invalidation. If set to -autoreload, TimesTen reloads the database after an invalidation. This is the default behavior.
-cachePolicy	Defines the policy used to determine when the cache agent for the database should run. always - Specifies that the cache agent should always be running for the database. This option immediately starts the cache agent and when the daemon restarts the cache agent is restarted. manual (default) - Specifies that the cache agent must be manually started and stopped. norestart - Specifies that the cache agent for the database is not to be restarted after a failure. 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 database 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 manages autorefresh cache groups and asynchronous writethrough cache groups. The cache administration user has extended privileges. See "Grant privileges to the Oracle database users" in the *Oracle TimesTen Application-Tier 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 database users" in the *Oracle TimesTen Application-Tier 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.

Option	Description
-cacheUidPwdSet	Sets the cache administration user name 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. ■ For all levels of DDLReplicationLevel, you can set the cache administration user name and password with the -cacheUidPwdSet option while the cache or replication agents are running. For more details, see "Changing user names or passwords used by replication" in the *Oracle TimesTen In-Memory Database Replication Guide*. ■ The cache administration user ID cannot be reset while there are cache groups on the database. The cache administration password can be changed at any time.
-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 loaded in system RAM only when in use (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.

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 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 starts. 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.
-repStart	Starts the database's replication agent.
-repStop	Stops the database's replication agent.
-stopTimeout *seconds*	Specifies that the TimesTen daemon should stop the cache agent if it does not 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

Some very performance sensitive applications use a database referred to by DSN SalesData. 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 keep the database in memory, 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 the following.

```
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, the database remains in RAM if applications are connected to the database. To set this policy, use:

```
ttAdmin -ramPolicy inUse -ramGrace 300 History
```

A database referred to by DSN `SalesData` contains data cached from an Oracle database. 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, you can use the `-cacheUidPwdSet` flag with the `-cacheUid` and `-cachePwd` options. 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 an Oracle database to a TimesTen database) or asynchronous writethrough cache groups, you must specify the cache administration user ID and password. This user account performs autorefresh and asynchronous writethrough operations.

To load data from an Oracle database, 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 TimesTen Application-Tier Database Cache User's Guide* for

more details. For details on other environment variables that you may want to set, see "Environment variables" in *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 11.2.2.0.0 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 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, use this utility to move databases from a TimesTen instance to a new TimesTen instance that is of the same major release, but of a different minor release. For example, you can move files from TimesTen 11.2.2.5.0 to TimesTen 11.2.2.6.0.

> **Note:** A major release refers to the first three digits of the release number. A minor release refers to the last two digits of the release number.

This utility is useful for testing a minor release of Times with an existing database. You can install the new release of TimesTen and move one or more databases to the new release without uninstalling the old TimesTen release.

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
```

Notes

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. For an overview of the TimesTen backup and restore facility, see "Migration, Backup, and Restoration" in *Oracle TimesTen In-Memory Database Installation 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, and you are creating a backup from a database other than the one previously backed up in the specified directory, 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 (default) - Performs a full file backup to the backup path specified by the *directory* and *fileprefix* parameters. 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

The ttBackup utility and the ttRestore utility backup and restore databases only when the first three numbers of the TimesTen release and the platform are the same. For example, you can backup and restore files between TimesTen releases 11.2.2.2.0 and 11.2.2.6.0. You cannot backup and restore files between releases 11.2.1.9.0 and 11.2.2.6.0. You can use the ttBulkcp or CS (UNIX only) utility to migrate databases across major releases or operating systems. You can use ttMigrate together with ttMigrateCS (client server version of ttMigrate) to migrate databases between 32- and 64-bit platforms or bit levels. You must use the -relaxedUpgrade option when

restoring data on a new bit-level. In the case of changing bit-levels, the database cannot be involved in a replication scheme. Follow the examples in "Moving a database between 32-bit and 64-bit platforms" in the *Oracle TimesTen In-Memory Database Installation Guide*.

When an incremental backup has been enabled, TimesTen creates a backup hold in the transaction log file. Call the `ttLogHolds` built-in procedure to see information about this hold. The backup hold determines 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 commit after the start of the backup operation 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.

Databases containing cache groups can be backed up as normal with the `ttBackup` utility. However, when restoring such a backup, special consideration is required as the restored data within the cache groups may be out of date or out of sync with the data in the back end Oracle database. See the section on "Backing up and restoring a database with cache groups" in the *Oracle TimesTen Application-Tier Database Cache User's Guide* for details.

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 (`ttBulkCp -i`), rows are copied into an existing TimesTen table from one or more ASCII files (or `stdin`).
- In copy-out mode (`ttBulkCp -o`), an entire TimesTen table is copied to a single ASCII output file (or `stdout`).

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] [-tsformat timeStampFormat]
[-dformat | -D dateFormat] [-F firstRow] [-L lastRow]
[-N ncharEncoding] [-Q 0|1] [-S errLevel] [-dateMode dateMode]
[-[no]tblLock]{-connStr connection_string | DSN}
[owner.]tableName [dataFile ...]

ttBulkCp -o [-sc] [-v 0|1] [-A 0|1] [-Cc | -Cnone]
[-nullFormat formatStr}
[-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 -C*c*	-Cnone disables the use of comments in the output file.-C*c* 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. For a list of legal fixed values, see "Date, time and timestamp values" on page 3-23. 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 database 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.
-h -help -?	Prints a short usage message and exits.
-helpfull	Prints a longer usage message and exits.
-i	Selects copy-in mode.
-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. This document refers to this mode as "no quote mode." 1 (default) - Indicates that strings should be quoted.This option overrides the QUOTES file attribute. This document refers to this mode as "quote mode."

Option	Description
`-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.
tableName	Specifies the name of the table to be saved or loaded. This parameter is case-insensitive.
`-tformat` *timeFormat*	Sets the time format. For a list of legal fixed values, see "Date, time and timestamp values" on page 3-23. The default value is `ODBC`. This option overrides the `TSFORMAT` file attribute. See also: `-D | -dformat` and `-tsformat`.
`-tsformat` *timestampFormat*	Sets the timestamp format. For a list of legal fixed values, see "Date, time and timestamp values" on page 3-23. 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` (default) - Prints a summary of rows copied upon completion.

Use the following options 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` (default) - `ttBulkCp` may write attribute lines into the output file. `1` - Suppresses output of attribute lines.
`-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 *Oracle TimesTen In-Memory Database Operations Guide*.

Option	Description
-nullFormat *formatStr*	Specifies the format in which NULL values are printed. Valid values are: null (default) - The word NULL is printed for null fields. empty - Nothing is printed for null fields. An empty LOB is printed as NULL in no-quotes mode and as " " in quote mode. When copied in, both NULL and " " are interpreted as a NULL LOB.
-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.

Use the following options in copy-in (-i) mode only. You must have INSERT privileges on the specified tables.

Option	Description
-cp *numTrans*	Sets the checkpoint policy for the copy in.
-cp final	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 (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 in which 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 "Data file format" on page 3-19), 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. Use this option (optionally 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
-m *maxErrors*	Specifies the maximum number of errors to report. The default is 1. If set to 0, ttBulkCp returns all error messages. There is no maximum limit.
-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 (sub-microsecond granularity). -S error (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, VARBINARY, BLOB, CLOB, or NLOB value is longer than its maximum column width. -t error (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.
-[no]tblLock	Specifies whether to use table-level or row-level locking, when copying rows into a TimesTen table. -tblLock - Indicates table-level locking. This is the default. -notblLock - Indicates row-level locking. For a single input stream into a table, using -tblLock is most efficient. Using -notblLock provides some performance benefit if you use multiple concurrent ttBulkCp sessions to insert into a single table in parallel.
-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 (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.

Option	Description
`-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`* processed rows. The default value is `1024`. Use the `-xp` option with the `-cp` option to enable periodic checkpointing of the database.

Data file format

This section describes the format the *`dataFile`* parameter.

Each line of a `ttBulkCp` input file is either 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 "File 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.
- File attribute lines are used for setting file attributes that control the formatting of the data file. Attribute lines begin with the ten-character sequence `##ttBulkCp`. The section "File attribute line format" on page 3-19 describes the full syntax for attribute lines. Attribute lines can appear anywhere in the data file.
- Data lines contain the rows of the table being copied. Data lines in the data file 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 section "Data line format" on page 3-21 describes the full syntax for data lines.

File 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.

File 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.

Most command line options take precedence over the values in the file attributes that are supported by `ttBulkCp`. The `CHARACTERSET` attribute is the only file attribute that overrides command line options.

The file attributes are:

- CHARACTERSET: Specifies the character set to be used to interpret the data file. If the file attribute is not set, the character set used to interpret the file is the one specified in the ConnectionCharacterSet connection attribute. For best performance, the value of the DatabaseCharacterSet connection attribute should match either the ConnectionCharacterSet connection attribute or this file attribute. If the character set supplied in ConnectionCharacterSet connection attribute or in this file attribute is different than the actual character set of the file, ttBulkCp may interpret data incorrectly.
- 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 database 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 data file.

- DFORMAT: Sets the date format. For a list of legal values, see "Date, time and timestamp values" on page 3-23. 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. For a list of legal values, see "Date, time and timestamp values" on page 3-23. 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. For a list of legal values, see "Date, time and timestamp values" on page 3-23. 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, CLOB, NCLOB: 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.
- \u*xyzw* (NCHAR and NVARCHAR2 only) The character whose unicode value is *xyzw*, where *xyzw* is a four-digit hexadecimal number, as in \ufe4a. The \u*xyzw* 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 usually, 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 data file. Trailing blanks are neither added nor removed.

Binary values

BINARY, VARBINARY, BLOB: 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 hexadecimal string is converted to a single binary byte, the hexadecimal string 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 data file.

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).
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.
FFn	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`.
+FFn	In copy-in mode, same as `+FF`. In copy-out mode, same as `.FF`*n*.

Specifier	Descriptions and restrictions
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: 2011-01-03 (default value)
Oracle	*DD-Mon-YYYY* Example: 03-Jan-2011
SYBASE1	*MM/DD/YYYY* Example: 01/03/2011
SYBASE2	*DD-MM-YYYY* Example: 03-01-2011
SYBASE3	*Mon*DD*YYYY* Example: Jan 03 2011

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: 2011-01-03 07:47:23
Oracle	*DD-Mon-YYYY*HH24:MI:SS+FF* Example: 03-Jan-2011 07:47:23

Format	Description
SYBASE1	*MM/DD/YYYY*HH24:MI:SS+FF* Example: 01/03/2011 07:47:23
SYBASE2	*DD-MM-YYYY*HH24:MI:SS+FF* Example: 03-01-2011 07:47:23
SYBASE3	*Mon*DD*YYYY*HH24:MI:SS+FF* Example: Jan 03 2011 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 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 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 irrecoverable error), use -m 0, as in:

```
ttBulkCp -i -e mytbl.err -m 0 mystore mytbl mytbl.dump
```

To ignore errors caused by constraint violations, use -d ignore, as follows.

```
ttBulkCp -i -e mytbl.err -d ignore mystore mytbl mytbl.dump
```

Notes

ttBulkCp explicitly sets the Overwrite connection attribute to 0, to prevent accidental destruction of a database. For more information, see "Overwrite" on page 1-44.

Real, float or double values may be rounded to zero when the floating point number is small.

The connection attribute PassThrough with a nonzero value is not supported with this utility and returns an error.

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 data file:

- 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. If there are no actual comments in the file, set the comment character to none to avoid characters from being misread as comment characters.

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.

WINDOWS

This utility is for use specifically with TimesTen tables. It is not supported with passthrough to an Oracle database.

On Windows, this utility is supported for all TimesTen Data Manager and Client DSNs.

It is recommended that you do not run DDL SQL commands while running ttBulkCp to avoid lock contention issues for your application.

See also

ttBackup
ttMigrate
ttRestore

ttCacheAdvisor

Description

Assists in configuring a TimesTen Cache for optimal performance and minimal storage overhead. This utility evaluates a captured SQL workload running on a target Oracle database, or an existing SQL tuning set. It also evaluates the schema definitions from the target database.

This utility analyzes the table and column usage patterns, and recommends TimesTen cache group definitions to improve the workload performance by generating a report and an implementation script. It also optionally evaluates the performance of the recommended cache and compares it with the performance of the target Oracle database.

Required privilege

This utility requires the instance administrator privilege.

Syntax

General analysis

```
ttCacheAdvisor
  -oraTarget
      {-oraConn Oracle_connection_string} ...
      -oraDirObject Oracle_directory_object
      [{-oraDirNfs path} | {-ftp network_connection_string}]
  -oraRepository
      -oraConn Oracle_connection_string
      -oraDirObject Oracle_directory_object
      [{-oraDirNfs path} | {-ftp network_connection_string}]
  -ttConn TimesTen_connection_string
  [-captureCursorCache minutes] [-flushSharedPool] [-plsqlInfo]
  [-cacheSize {{size{MB|GB|TB}} | UNLIMITED}]
  [-writethruThreshold percent%]
  [-report path]
  [-evalSqlPerf [maximum[%]]]
  [-task task_name]
  [-description string]
  [-rerun]
  [-showSql]
  [-trace [traceName]]
  [{-import [-noSchema] path} ...]
  [-add
    {-tableAttrib [owner.]table_name
      {-pk '(column_name, ...)' |
       -fk '(column_name, ...) REFERENCES (primary_key_column_name, ...)
         [ON DELETE CASCADE]' |
       -where '(predicate)'} ...
    } ...]
  [-drop
    {-tableAttrib [owner.]table_name
      {-pk | -fk | -where} ...
    } ...]
  [{{-add | -drop} -sqlSet [owner.]sql_set_name}]
  [{-command | @} path]
```

You can repeat the options for `-oraTarget` to specify multiple users for the same target Oracle database to analyze objects from more than one schema. There can be only one Oracle repository specification in a Cache Advisor evaluation run.

String values may be optionally enclosed in single or double quotes. If the string begins with a dash (-), then the string *must* be enclosed in single or double quotes.

Print a usage message and exit.

```
ttCacheAdvisor {{-help | -h} [option]}
```

Print the release number of `ttCacheAdvisor` and exit.

```
ttCacheAdvisor {-V | -version}
```

Export a workload and schema definitions from the target Oracle database to files:

```
ttCacheAdvisor
  [{-command|@} path]
  {-oraTarget
     -oraConn oracle_connection_string
     -oraDirObject oracle_directory_object
     [{-oraDirNfs| -ftp} network_connection_string]
  } ...
  -captureCursorCache minutes [-flushSharedPool]
  -export [-zip] path
  [-showSql]
```

Display a summary or details of existing tasks in the repository Oracle database:

```
ttCacheAdvisor  [{-command|@} path]
  -oraRepository -oraConn oracle_connection_string
  -showTask [task_name]
  [-showSql]
```

Drop an existing task from the repository Oracle database:

```
ttCacheAdvisor
  [{-command|@} path]
  -oraRepository -oraConn oracle_connection_string
  -dropTask task_name
  [-showSql]
```

For more information about using Cache Advisor to capture and analyze a SQL workload running on a target Oracle database, see *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Options

`ttCacheAdvisor` has the options:

Option	Description
`{-add | -drop}`	`-add` adds the specified item to the `ttCacheAdvisor` task. `-drop` removes the specified item from the `ttCacheAdvisor` task. `-drop` is supported only with the `-rerun` option. Use these options to add or drop cache group and cache table attributes, and workload options.

Option	Description
`-cacheSize {{`*size*`{MB|GB|TB}} | UNLIMITED}`	Specifies the maximum TimesTen database size in megabytes (`MB`), gigabytes (`GB`) or terabytes (`TB`). The default is `UNLIMITED`. The value should be set to amount of the permanent memory available on the system where you plan to deploy TimesTen Cache.
`-captureCursorCache` *minutes*	Specifies that `ttCacheAdvisor` capture a SQL workload from the Oracle cursor cache on the Oracle target database. The capture operation occurs as a series of snapshots and lasts for the specified number of minutes. Fractions of minutes can be specified using a decimal point value for *minutes*. The Oracle target database manages its cursor cache by unloading cursors for specific SQL statements based upon internal heuristics. Therefore, it is not possible to predict exactly which cursors for which SQL statements are in the Oracle cursor cache when a given cursor cache snapshot is captured. Oracle recommends a capture duration of at least 10 minutes to avoid having some SQL statements not being captured. However, certain application workload behaviors or a small Oracle cursor cache may require a longer capture duration to avoid SQL statements not being captured.
`{-command|@}` *path*	Specifies a file of `ttCacheAdvisor` commands. You can use this option anywhere on the command line that an option is allowed. Command files can be nested to any depth if they are not recursive. Blank lines and lines that start with the `#` comment delimiter are ignored in the files.
`-description` *string*	Specifies a description of the `ttCacheAdvisor` task. The description can be up to 4000 bytes in length.
`-dropTask` *task_name*	Drops the specified task, including SQL tuning sets created or copied on behalf of the task that are not being used by any other tasks.
`-evalSqlPerf [`*max*`[%]]`	Specifies that `ttCacheAdvisor` compare the performance of the SQL workload statements in the TimesTen database with the target Oracle database. The `-evalSqlPerf` option should only be used on a test target database and not on a production target database. You can specify a number or a percentage of SQL statements to evaluate. Include the `%` character for a percentage. The default is 100%. If you limit the number of SQL statements to evaluate, then `ttCacheAdvisor` evaluates the SQL statements that appear to have the greatest potential for caching based on the captured workload and target schema statistics. When using the `-evalSqlPerf` option, you must specify a direct connect DSN with the `-ttConn` option and not a client DSN.

Option	Description
`-export [-zip] path`	Specifies that `ttCacheAdvisor` export the workload captured with the `-captureCursorCache` option. The -export option cannot be used with the `-oraRepository`, `-ttConn`, `-report` nor `-task` options. The `-zip` option compresses the exported workload into a single zip file. The `-zip` option requires the `zip` utility to be installed on the TimesTen system and be in the executable path. *path* specifies the location on the TimesTen system for the export files. If the `-zip` option is specified, then `ttCacheAdvisor` creates a single compressed file with the `.zip` file extension whose name is the last string of *path*. If the `-zip` option is not specified, then `ttCacheAdvisor` writes the export files to the *path* directory. The prefix of the names of all files created by the `-export` option is the last string of *path*.
`-fk '(`*column_name*`, ...) REFERENCES (`*primary_key_column_name*`, ...) [ON DELETE CASCADE]'`	Specifies a foreign key for the table. If the specified foreign key creates a cycle with an existing foreign key in the Oracle database, then `ttCacheAdvisor` uses the specified foreign key and not the existing foreign key. '(*column_name*, ...) `REFERENCES` (*primary_key_column_name*, ...) `[ON DELETE CASCADE]'` is required when the `-fk` option is used with the `-add` option, and is optional when used with the `-drop` option.
`-flushSharedPool`	Flushes the shared pool on the target Oracle database before capturing a workload. Can only be specified with the `-captureCursorCache` option. Use the `-flushSharedPool` option only with a test target database that may contain SQL statements in the cursor cache that are not associated with the application being analyzed. Do not use the `-flushSharedPool` option with a production target database.
`{-help | -h} [`*option*`]`	Prints a usage message and exits.
`-import [-noSchema] path`	Specifies that `ttCacheAdvisor` import workloads and schemas that have previously been exported. The -import option cannot be used with the `-captureCursorCache` nor `-flushSharedPool` options. Use `-noSchema` to import a workload without its schema. This option is useful when the target schema is already defined in the Oracle repository database. *path* specifies the location of the exported workloads and schemas on the TimesTen system. The `-import` option first searches for a directory named *path* that contains the export files and then for a file named *path.zip*. The `-import` option can be specified multiple times for the same `ttCacheAdvisor` execution to import multiple previously exported schemas or workloads. If you exported the workloads and schema using the `-zip` option, the `unzip` utility must be installed on the TimesTen system and be in the executable path.

Option	Description
-oraConn *Oracle_connection_string*	Specifies the Oracle database user and optional password, Oracle Net Service Name, and any optional connection attributes. The standard format of an Oracle connection string is: *Oracle_user*[/*password*]@Oracle_net_service_name ttCacheAdvisor prompts for the password if it is omitted from the Oracle database connection string. If asterisk (*) is specified for the password, then ttCacheAdvisor assumes that automatic authorization on the Oracle database has been configured and does not prompt for the password. No more than one connection to the repository Oracle database can be specified. Multiple connections can be specified to the Oracle target database to analyze the schema definitions from multiple target schemas and to facilitate performance evaluation of SQL statements that were originally executed by different Oracle target database users. To specify multiple connections to the Oracle target database, specify the -oraConn option multiple times after the -oraTarget option using different target Oracle database users and the same target Oracle net service name. ttCacheAdvisor uses the first -oraTarget -oraConn connection specified to capture the workload if -captureCursorCache is specified.
-oraRepository	Specifies the repository Oracle database (version 11.2.0.2 or later) used by ttCacheAdvisor to perform analysis of application workload and schema definitions. If either -oraRepository or -oraTarget is not specified, then the same Oracle database is used as the repository and target.
-oraTarget	Specifies the target Oracle database (version 10.2.0.4 or later) to cache in a TimesTen database. If either -oraTarget or -oraRepository is not specified, then the same Oracle database is used as the target and repository.
-pk '(*column_name*, ...)'	Specifies a primary key for the table. This overrides any primary key on the Oracle database table. '(*column_name*, ...)' is required when the -pk option is used with the -add option, and is optional when used with the -drop option.
-plsqlInfo	Augments the captured workload with information about the PL/SQL objects from which SQL statements originated. Can only be specified with the -captureCursorCache option. The workload augmentation process may increase the load on the target database.
-report *path*	Specifies a directory on the TimesTen system where the ttCacheAdvisor writes the recommendation report and implementation script. The default is the task_name directory where ttCacheAdvisor was invoked. The specified directory cannot exist as the ttCacheAdvisor utility creates the directory before writing files into it that constitute the report.
-rerun	Specifies that ttCacheAdvisor rerun the current task. The -rerun option must be used with the -task *task_name* option.

Option	Description
-showSql	Specifies that ttCacheAdvisor display the SQL statements it executes on the TimesTen database, the target Oracle database and the repository Oracle database. ttCacheAdvisor prompts for continuation after displaying the statements. Use this option to assess the impact of ttCacheAdvisor before incurring the impact.
-showTask [*task_name*]	Displays information about the specified task or all tasks. If *task_name* is not specified, then summary information is displayed about all tasks stored in the repository Oracle database. If *task_name* is specified, then detailed information is displayed about the specified task.
-sqlSet [*owner.*]*sql_set_name*	Specifies a SQL tuning set to add or drop from the ttCacheAdvisor task. When -add is specified, ttCacheAdvisor looks for the specified SQL tuning set first on the repository Oracle database and then on the target Oracle database. When -drop is specified, ttCacheAdvisor drops the SQL tuning set from the task. If the SQL tuning set was created or copied on behalf of the task, then ttCacheAdvisor also permanently drops the SQL tuning set from the repository database. You can add or drop multiple SQL sets on the command line. For example: -add -sqlSet ADVISOR.TEST1 -add -sqlSet ADVISOR.TEST2
-tableAttrib [*owner.*]*table_name*	Specifies the cache group attributes and cache table attributes for ttCacheAdvisor to use for caching recommendations. The schema and table name can include a % wildcard character. When dropping an attribute, the schema and table name must match what was specified when they were added, including any wildcard characters. Using a wildcard character for the schema or table name, you can specify an attribute to apply to more than one table. ttCacheAdvisor uses the first specified value for a specific table.
-task *task_name*	Specifies a task name. The name can be up to 30 bytes in length and must be unique among ttCacheAdvisor tasks stored in the repository Oracle database. In other words, no task with the name *task_name* can exist at the time you run the ttCacheAdvisor utility. Otherwise, the task must first be dropped using the -dropTask option. A task is an object that contains information about the workload, performance results and ttCacheAdvisor options specified by the user. Use the task name to run, rerun, show or drop the task. The default task name is *user-name_host-name_timestamp*, truncated to 30 bytes if necessary.

Option	Description
-trace [*traceName*]	Captures debug trace information that can be provided to Oracle TimesTen Customer Support. By default, all trace files are placed in the directory: *install_dir*/info/advisor/*traceyyyymmddhhmmss* If you provide a trace name with the -trace *traceName* option, the trace files are placed in the directory: *install_dir*/info/traceName/*traceyyyymmddhhmmss* *yyyymmddhhmmss* is a timestamp that is set when the Cache ttCacheAdvisor starts and includes a four-digit year, two-digit month, two-digit day, two-digit hour, two-digit minute and two-digit second. If the zip executable exists in any of the directories set in the PATH environment variable, the trace files are automatically compressed into a single compressed file named ttca_debug_log.zip, using the first zip executable encountered in the path. (Other compression utilities are not recognized or used.)
-ttConn *TimesTen_connection_string*	Specifies a TimesTen database for ttCacheAdvisor to use for cache validation and evaluation. *TimesTen_connection_string* is a TimesTen connection string that contains at least the DSN attribute and the UID attribute. Set the UID attribute to the TimesTen cache manager user. Example: "DSN=ttca;UID=user" ttCacheAdvisor prompts for the TimesTen cache manager user's password if the PWD attribute is omitted from the connection string. ttCacheAdvisor prompts for the Oracle database cache administration user's password if the OraclePWD attribute is omitted from the connection string and the -evalSqlPerf option is specified. ttIsql must be able to connect to the TimesTen database using a direct connection from the host where ttCacheAdvisor is invoked. The TimesTen database must be set up for use by TimesTen Cache as described in the chapter "Using the Cache Advisor" of the *Oracle TimesTen Application-Tier Database Cache User's Guide*before invoking ttCacheAdvisor.
-V | -version	Prints the release number of ttCacheAdvisor and exits.
-where '(*predicate*)'	Specifies a WHERE clause for the table if using a readonly cache group. TimesTen ignores the value of this option for an AWT cache group. The format of *predicate* is any valid TimesTen WHERE clause that can be used in cache table definitions. '(*predicate*)' is required when the -where option is used with the -add option, and is optional when used with the -drop option.

Option	Description
`-writethruThreshold` *percent%*	Specifies a threshold of update or write operations, expressed as a percentage, including the `%` character as a suffix. If the percentage of update operations in the workload on a table or related set of tables is less than or equal to the specified threshold, then `ttCacheAdvisor` recommends a read-only cache group to cache the Oracle database tables. Otherwise, `ttCacheAdvisor` recommends an asynchronous writethrough (AWT) cache group. The default threshold is zero which means that `ttCacheAdvisor` recommends an AWT cache group if the workload contains at least one update operation on a table or related set of tables.

Advanced Options

The following options are recommended for advanced Cache Advisor users. You can use these options for improved file transfer performance with the `-export` and `-import` options:

Option	Description
`-ftp` *network_connection_string*	Specifies a network connection for transferring files between the target Oracle database, repository Oracle system and TimesTen database. *network_connection_string* is specified as `[`*user_name*`][@`*host_*`name]`. *user_name* defaults to the name of the operating system user that runs `ttCacheAdvisor`. *host_name* defaults to the Oracle target host name when used with the `-oraTarget` option, and the Oracle repository host name when used with the `-oraRepository` option. If anonymous ftp access has been provided, then no password is required. Otherwise, `ftp` prompts for a password whenever `ttCacheAdvisor` needs to access a host to copy a set of files to another host. By default, Cache Advisor uses the OCI connection specified with the `-oraConn` option to transfer files. When using either the `-export` or `-import` options, the `-ftp` option can improve file transfer performance. This performance advantage can be offset by password prompting if anonymous `ftp` access is not enabled. To use the `-ftp` option, you must configure the directory for use with the `-ftp` option. You must precede the `-ftp` option with either the `-oraTarget` or `-oraRepository` option to indicate whether it applies to the target or repository.

Option	Description
-oraDirNfs *path*	Specifies a network connection for transferring files between the target Oracle database system, repository Oracle database system, and TimesTen database system. path is the directory path on the TimesTen system where Advisor is invoked that is network mounted to the directory path corresponding to the Oracle directory object specified by the -oraDirObject option. You must precede the -oraDirNfs option with either the -oraTarget or -oraRepository option to indicate whether it applies to the target or repository. By default, Cache Advisor uses the OCI connection specified with the -oraConn option to transfer files. When using either the -export or -import options, the -oraDirNfs option can improve file transfer performance. To use the -oraDirNfs option, you must configure the directory for use with the -oraDirNfs option.
-oraDirObject *Oracle_directory_object*	Specifies an Oracle database directory object that ttCacheAdvisor uses to export and import workload and schema information to and from dump files using Data Pump. The directory object must reference a directory path in a local file system on the target Oracle database host when used with the -oraTarget option, or a directory path in a local file system on the repository Oracle database host when used with the -oraRepository option. The default directory object is TTCA_DIRECTORY, which is created during Cache Advisor setup. The TTCA_DIRECTORY directory object is not configured for use with the -oraDirNfs or -ftp options. Use the -oraDirObject option when you use either the -oraDirNfs or -ftp options. You must precede the -oraDirObjec option with either the -oraTarget or -oraRepository option to indicate whether it applies to the target or repository.

Examples

Capture a SQL workaround on the target database for a duration of 30 minutes. Flush the shared pool on the target database before capturing the workload. Analyze the workload and produce a report in the ./testtask directory.

```
% ttCacheAdvisor -oraTarget -oraConn orauser@targetdb \
-oraRepository -oraConn advisor@repositorydb \
-ttConn "DSN=cacheadvisor;UID=cacheuser" \
-captureCursorCache 30 -flushSharedPool -task testtask
```

Capture a SQL workaround on the target database for a duration of 15 minutes. Evaluate the performance of all SQL statements executed in the TimesTen database and in the target database. Write the HTML report files to the /home/ttuser/cacheadvreport directory.

```
% ttCacheAdvisor -oraTarget -oraConn orauser@targetdb \
-oraRepository -oraConn advisor@repositorydb \
-ttConn "DSN=cacheadvisor;UID=cacheuser" \
-report /home/ttuser/cacheadvreport -captureCursorCache 15 -evalSqlPerf
```

Capture a SQL workaround on the target database for a duration of 10 minutes. Flush the shared pool on the target database before capturing the workload. Export the captured workload into a single file named cacheadvcapture.zip written to the /home/ttuser directory.

```
% ttCacheAdvisor -oraTarget -oraConn orauser@targetdb \
 -captureCursorCache 10 -flushSharedPool -export -zip \
/home/ttuser/cacheadvcapture
```

Import and analyze the SQL workload from the `/home/ttuser/cacheadvcapture.zip` file. Write the HTML report files to the `/home/ttuser/cacheadvreport` directory.

```
% ttCacheAdvisor -oraTarget -oraConn orauser@targetdb \
-oraRepository -oraConn advisor@repositorydb \
-ttConn "DSN=cacheadvisor;UID=cacheuser" \
-report /home/ttuser/cacheadvreport -task temptask \
-import /home/ttuser/cacheadvcapture
```

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: You should always enclose directory and file names in double quotes, 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]
ttCapture [-noinstinfo] [-nosysinfo] [-stdout | -dest dir] [-logdir dir]
 [-noconnect] [dspath | DSN]
ttCapture -noconnect [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`	Specifies 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/2011.ds0`, `/home/payroll/2011.ds1`, and several transaction log files `/home/payroll/2011.log`*`n`*, *`dspath`* is `/home/payroll/2011`. NOTE: The *`DSN`* and *`dspath`* options are mutually exclusive. If you do not supply either option, `ttCapture` does not provide any database information.

Option	Description

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-logdir` *dir*	Specifies the location of the log directory. Must be used with the `-dspath` option. If not specified, the log directory may not be available.
`-noconnect`	Specifies that the utility should capture information on the DSN without connecting to it. If specified, some information, such as `ttConfiguration` output and replication schemes, is not included in the output. This option is useful if you do not want to load a large database or if you are reporting a problem where connections are failing.
`-noinstinfo`	Indicates that `ttCapture` should not capture any installation information.
`-nosysinfo`	Indicates that `ttCapture` should not capture any system information.
`-stdout`	On UNIX systems, `ttCapture` writes all output to `stdout`, instead of writing the output to a file. On Windows, `ttCapture` writes to a `Command` prompt.
`-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
```

Notes

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`* connection attribute in the database's DSN. For example, for a database consisting of files `/home/payroll/2011.ds0`, `/home/payroll/2011.ds1`, and several transaction log files `/home/payroll/2011.log`*`n`*, *`dspath`* is `/home/payroll/2011`.
`-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*	Indicates 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 (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 (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 the following.

```
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.

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` enables 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 Replication Guide*.

These commands are available only with advanced high availability:

- `ttCWAdmin -addMasterHosts`
- `ttCWAdmin -addSubscriberHosts`
- `ttCWAdmin -createVIPs`
- `ttCWAdmin -delMasterHosts`
- `ttCWAdmin -delSubscriberHosts`
- `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 administrator 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 [-timeout seconds] -dsn DSN

ttCWAdmin -relocate -dsn DSN

ttCWAdmin -reauthenticate -dsn DSN

ttCWAdmin -ocrConfig

ttCWAdmin -beginAlterSchema -dsn DSN

ttCWAdmin -endAlterSchema -dsn DSN

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 -start [-noapp] -dsn DSN

ttCWAdmin -stop -dsn DSN

ttCWAdmin -startapps -dsn DSN

ttCWAdmin -stopapps -dsn DSN

ttCWAdmin -shutdown [-hosts "host_name1, host_name2[, ...]"]
```

Options

`ttCWAdmin` has these options:

Option	Description
`-addMasterHosts`	Adds spare hosts to the pool 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 pool 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.
`-beginAlterSchema`	Enables manual alteration, addition or dropping of cache groups to the active standby pair replication scheme when automatic include of new schema objects in the active standby pair scheme is not possible. Also, enables creation of PL/SQL procedures, sequences materialized views and indexes on tables with data. Enables addition of a read-only subscriber that is not managed by Oracle Clusterware. While adding objects to the schema, the active standby pair is brought down, but the active node stays attached if you are using grid. See also: `-endAlterSchema`.

Option	Description
-create	Creates the active standby pair replication scheme for the specified DSN and creates the associated action scripts. This command: ■ Prompts for the name of a TimesTen internal user with ADMIN privileges. TimesTen uses this internal user to create the active standby pair. If cache groups are being managed by Oracle Clusterware (if the attribute CacheConnect=Y in the cluster.oracle.ini), enter the TimesTen cache manager user name. ■ Prompts for the TimesTen password for the previously entered user name. ■ If cache groups are being used, prompts for the password for the Oracle database user that has the same name as the cache manager. This password is provided in the OraclePWD connection attribute when the cache manager user connects. This Oracle database user is used to set the autorefresh states for cache groups. ■ Prompts for a random string used to encrypt the above information.
-createVIPs	Creates virtual IP addresses for the active standby pair. If no DSN is specified, displays the information of all active standby pairs managed under the same TimesTen instance administrator and TimesTen instance name managed by Oracle Clusterware.
-delMasterHosts	Deletes spare hosts to the pool 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.
-delSubscriberHosts	Deletes spare hosts to the pool 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.
-endAlterSchema	Issued this option after an operation using the -beginAlterSchema option. Rolls out the active standby pair after objects have been added to the schema, while recording the new replication checksum. The old standby is being destroyed and recreated through duplicate
-h -help -?	Prints a usage message and exits.
-init	Starts the TimesTen cluster agent.

Option	Description
-ocrConfig	TimesTen cluster information is stored in the Oracle Cluster Registry (OCR). This option registers the admin user in the OCR. You must register the admin user once before performing any of the cluster initialization steps. On UNIX and Linux systems, login as the root user and run this command from any host in the system before creating any clusters. On Windows systems, login as the instance administrator to run this command. You do not need to perform this step when starting an existing cluster that you have shutdown.
-reauthenticate	This command reauthenticates the user names and passwords after any of them have been modified. Even if only a single password is changed, this command still prompts for all user names and passwords. ■ Prompts for the name of a TimesTen internal user with ADMIN privileges. TimesTen uses this internal user to create the active standby pair. If cache groups are being managed by Oracle Clusterware (if the attribute CacheConnect=Y in the cluster.oracle.ini), enter the TimesTen cache manager user name. ■ Prompts for the TimesTen password for the previously entered user name. ■ If cache groups are being used, prompts for the password for the Oracle database user that has the same name as the cache manager. This password is provided in the OraclePWD connection attribute when the cache manager user connects. This Oracle database user is used to set the autorefresh states for cache groups. ■ Prompts for a random string used to encrypt the above information. For more details, see "Changing user names or passwords when using Oracle Clusterware" in the *Oracle TimesTen In-Memory Database Replication Guide*.
-relocate	Relocates the database from the local host to the next available spare host specified in the MasterHosts attribute in the cluster.oracle.ini 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 High Availability (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 [-noapp]	Starts the cluster active standby pair. This results in starting all of the agents on the active database, creation of the standby database and the subscriber databases (if they exist) through duplicate if necessary, and subsequent starting of all agents on those databases. If you specify -noapp, the applications are not started. You can use the -startapps option to start the applications later.

Option	Description
-startapps	Starts the applications in the cluster.
-stopapps	Stops the applications in the cluster.
-status	Obtains the status of resources in the cluster.
-stop	Stops the replication agent and the cache agent and disconnects the application from both databases of an active standby pair. Also automatically detaches a grid member from a cache grid that is managed by Oracle Clusterware.
-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.
-timeout *seconds*	Specifies a timeout value for the -switch option. Specify an integer value greater than 0. The default is 900 seconds. If you enter an invalid value, TimesTen uses the default value of 900 seconds. If the timeout expires, TimesTen returns an error message and fails to verify 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
```

Notes

When you use Oracle Clusterware with TimesTen, you cannot use these commands and SQL statements:

- CREATE ACTIVE STANDBY PAIR, ALTER ACTIVE STANDBY PAIR and DROP ACTIVE STANDBY PAIR SQL statements.
- The -cacheStart and -cacheStop options of the ttAdmin utility after the active standby pair has been created.
- The -duplicate option of the ttRepAdmin utility.
- The ttRepStart and ttRepStop built-in procedures.

- Built-in procedures for managing a cache grid when the active standby pair in a cluster is a member of a grid.
- The `-repStart` and `-repStop` options of the `ttAdmin` utility.

In addition, do not call `ttDaemonAdmin` -stop before calling `ttCWAdmin -shutdown`.

The TimesTen integration with Oracle Clusterware accomplishes these operations with the `ttCWAdmin` utility and the attributes in the `cluster.oracle.ini` file.

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 | -restartserver]
ttDaemonAdmin [-force] -stopserver
ttDaemonAdmin -verbose
```

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.
`-restartserver`	Restarts the TimesTen Server.
`-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.

Use the `-force` option 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 includes stopping the replication agent and the cache agent, if they are running. This decreases startup time when the daemon is restarted. In addition, not stopping application connections or agents can result in the database becoming in validated.

If the Oracle Clusterware 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 Clusterware agent, the main daemon stops temporarily with this command, but then restarts. To stop the Oracle Clusterware 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

The TimesTen daemon (referred to as the TimesTen Data Manager Service on Windows) and its subdaemons and agents write error and status messages to the following daemon logs:

- A user error log that contains information you should be aware of, including 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 enables you to do the following:

- Control the types of events and categories of messages that are reported in the user error log.
- Display all messages or selected categories of messages from the log to the standard output.

Required privilege

This utility requires the instance administrator privilege.

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]
[-file filename] [-facility name]
[-n computer]
```

Notes:

- The `-file` and `-facility` options apply only on UNIX.
- The `-n` option applies only on Windows and is not relevant in typical usage.

Options

`ttDaemonLog` has the options:

Option	Description
`-b`	Prints all TimesTen-generated log entries.
`-f`	When the end of the log is reached, `ttDaemonLog` does not terminate but continues to execute, periodically polling the 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. **Note:** This option applies only on UNIX.

Option	Description
`-file` *filename*	Specifies the file into which TimesTen logs messages. If not specified, examine the system's `syslog` configuration to determine where TimesTen messages are being logged. **Note:** This option applies only on UNIX.

`-h`
`-help`
`-?`
	Shows `ttDaemonLog` usage information and exits.

-maxlines	Maximum number of lines at end of the log to display Defaults to `40` lines if `-f` is specified. If `0` is specified, there is no maximum.

`-logcomponent` *component*
`-nologcomponent` *component*
	By default, all categories of messages are logged, but you can use `-logcomponent` to specify a category to be logged, or `-nologcomponent` to specify a category to not be logged. You can specify only a single component, but can run `ttDaemonLog` with these options multiple times to determine the desired set of messages.
	If a DSN or connection string is specified, the option applies only to the specified database. You can run `ttDaemonLog` multiple times to set these options for multiple databases.
	Supported categories are:
	`ALL` (default): For all messages.
	`CACHE`: For messages from the cache agent, designated by `CAC`
	`DAEMON`: For messages from the main daemon and subdaemons
	`DAEMONDBG`: For additional information from the main daemon and subdaemons
	`REPLICATION`: For messages from the replication agent, designated by `REP`
	`SERVER`: For messages from TimesTen Server

`-loglevel` *level*	Specifies a cutoff for the level of messages that are logged in the support log. A lower value results in fewer messages. (For example, if you specify level 5, messages of level 1, 2, 3, 4, or 5 would be logged.) This option is typically relevant only for Customer Support use. If a DSN or connection string is specified, the option applies only to that database.
`-logreset`	Resets event logging parameters.
`-msg` *messagestring*	Inserts the specified text 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. **Note:** This option applies only on Windows and only if you are using the Windows Event Log for TimesTen logging, which is not typical usage.
`-r`	Prints only the TimesTen replication agent log. (Same as `-show replication`.)
`-s`	Prints only the TimesTen Server log. (Same as `-show server`.)
`-setverbose` `-setquiet`	Enables (`-setverbose`) or disables (`-setquiet`) TimesTen verbose logging.

Option	Description
`-show` *type*	When you use `ttDaemonLog` to display log messages to the standard output, you can use the `-show` option with one of the following types to limit the displayed log messages to that type only: `all` (default): Shows all messages. `replication`: Shows only log messages from replication agents. (Same as `-r` option.) `cache`: Shows only log messages from cache agents. `server`: Shows only log messages from TimesTen Server. (Same as `-s` option.) **Note:** You cannot show a category whose logging has been disabled through `-[no]logcomponent`.
`-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. This option can be applied to selected databases or all databases.

To display all the output from the TimesTen daemon and server on your local computer:

```
ttDaemonLog
```

To prevent messages and errors related to replication for all databases from being written to the log:

```
ttDaemonLog -nologcomponent replication
```

To prevent messages and errors related to replication for the `masterdsn` database from being written to the log:

```
ttDaemonLog -nologcomponent replication masterdsn
```

To prevent both replication and TimesTen Cache errors and messages from being written:

```
ttDaemonLog -nologcomponent replication
ttDaemonLog -nologcomponent cache
```

If, after disabling a component through the `-nologcomponent` option, you want to re-enable it, you can use the `-logcomponent` option. For example, after disabling messages for replication and TimesTen Cache as shown in the preceding example, you can re-enable replication messages as follows:

```
ttDaemonLog -logcomponent replication
```

To re-enable logging for all TimesTen components, use the `-logreset` option:

```
ttDaemonLog -logreset
```

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:

```
ttDaemonLog -show server
```

To display just the replication agent messages:

```
ttDaemonLog -show replication
```

To display just the cache agent messages:

```
ttDaemonLog -show cache
```

To display all messages from the TimesTen processes:

```
ttDaemonLog -show all
```

To restore logging to its default "verbose" level, use the `-setverbose` option:

```
ttDaemonLog -setverbose
```

On UNIX, to write the log output to the file `/var/adm/syslog/syslog.log`:

```
ttDaemonLog -file /var/adm/syslog/syslog.log
```

On UNIX, to direct logging to the `local7` facility:

```
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, change the options in the `ttendaemon.options` file. 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/2011.ds0`, `/home/payroll/2011.ds1`, and several transaction log files `/home/payroll/2011.log`n, *dspath* is `/home/payroll/2011`.

`-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.

Examples

```
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 database objects from autorefresh or AWT cache groups. If there are autorefresh or AWT cache groups in the database, execute the `cachecleanup.sql` script to clean up the cache objects in the Oracle database for that particular database, 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`*.

The `ttIsql` command attempts to cancel an ongoing ODBC function when the user presses `Ctrl-C`.

On UNIX, this utility is supported for TimesTen Data Manager DSNs. Use `ttIsqlCS` for client/server DSNs.

The `ttIsql` utility starts with `AUTOCOMMIT` turned on, even when running a script. You can turn `AUTOCOMMIT` off and back on as necessary.

For more details on the `ttIsql` utility, see the chapter "Using the ttIsql Utility" in the *Oracle TimesTen In-Memory Database Operations Guide*

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 semicolon separated list of `ttIsql` commands to execute on startup.
`-f` *`filename`*	Read SQL statements 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.

Option	Description
-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.
-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 (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 3-72.

Boolean commands can accept the values "ON" and "OFF" or "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 validates for that type. The default (enclosed in quotes) is assigned if the user just presses enter. The prompt is 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.

Command	Description
`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.
`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 `waitfor` 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.

Command	Description
bye exit	Exits ttIsql.
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. If no value is specified, displays the current value.
cachesqlget [ASYNCONOUS_WRITETOUGH | 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 database objects associated with a readonly 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 database 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 database 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.
compare *varA VarB*	Compares the values of two variables and reports if they are different. The first difference is reported.

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.
createandloadfromoraquery [*owner_name.*]*table_name* [*num_threads*] *query*	Takes a table name, the number of threads for parallel load and an Oracle SELECT statement. Creates the table in TimesTen if the table does not exist. Then, loads the table with the query result from the Oracle database. If the command creates the table, the table column names and types are derived from the query result. **Notes:** ■ The specified TimesTen table cannot be a system table, a synonym, a view, a materialized view or a detail table of a materialized view, a global temporary table or a cache group table. ■ The query cannot have any parameter bindings. ■ Any unsupported column types result in a warning being logged. The output issues a comment for the unsupported column data type. ■ If you do not supply a value for num_threads, defaults to four threads. ■ For details and usage information, see "Loading data from an Oracle database into a TimesTen table" in the *Oracle TimesTen In-Memory Database Operations Guide*. ■ You must rollback or commit after executing this operation. ■ Also see the NOTES section in the description of the built-in procedure ttLoadFromOracle. **Required Privileges:** Requires INSERT privilege on the table specified. Also, requires the CREATE TABLE privilege if the table does not exist. The Oracle session user must have all required privileges to execute the query on the Oracle database.
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 3-72.

Command	Description
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, cache groups, PL/SQL functions, PL/SQL procedures, PL/SQL packages 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. When describing cache groups, reports information on cache groups defined in the currently connected data source, including the state of any terminated databases that contain autorefresh cache groups. 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|g|t]	Prints database size information in KB, MB, GB or TB. The default is KB. The output indicates the unit returned.
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
edit [*file* | !*history_search_command*]	You can use the ttIsql edit command to edit a file or edit ttIsql commands in a text editor. The ttIsql edit command starts a text editor such as emacs, gedit, or vi. If TimesTen does not find an exact file match for the specified *file* parameter, it searches for *file*.sql. If neither file exists, ttIsql starts the editor with the file file. You can edit a SQL statement that is stored in the history list of the current ttIsql session. When calling the ttIsql edit command specify the ! character followed by the number of the command or a search string. If you execute the ttIsql edit command with a history_search_command parameter, ttIsql executes the contents of the file after you exit the text editor. The contents of the file are executed as a single ttIsql command. If you do not want to execute the contents of the file, delete the contents of the file and save the file before you exit the editor. You can only use one parameter at a time. The history_search_command parameter is defined as the ! character followed by the number of the command or a search string. If you do not specify a ! character, the ttIsql edit command interprets the parameter as file. If you do not specify a parameter or specify !!, the last ttIsql command is edited. You can specify the default editor by defining the ttIsql _EDITOR define alias. The following example sets the default editor to vi: `Command> DEFINE _EDITOR=vi` If you do not define the _EDITOR define alias, ttIsql uses the editor specified by the VISUAL environment variable. If the _EDITOR define alias and the VISUAL environment variables are not set, ttIsql uses the editor specified by the EDITOR environment variable. When _EDITOR, VISUAL, and EDITOR are not set, vi is used for UNIX and notepad.exe is used for Windows. For more details, see "Using the ttIsql edit command" in the *Oracle TimesTen In-Memory Database Operations Guide*.
exec [*connect_id.*] *command_id*] | *PLSQLSTMT*	Executes the prepared command *command_id* o n 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.

Command	Description
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.
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 do not 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.

Command	Description
`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. To see the exit status of the command, use the `define` command with `_EXIT_STATUS`.
`if-then-else`	The `if-then-else` command construct enables you to implement conditional branching logic in a `ttIsql` session. For more details, see "Syntax for the IF-THEN-ELSE command construct" on page 3-69.
`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 [optional_monitor_column]`	Formats the contents of the `SYS.MONITOR` table for easy viewing. If the *optional_monitor_column* is specified, only that column is displayed.
`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.
`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 `REF CURSOR`, then the results are fetched and printed.

Command	Description
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 describes 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 TimesTen Cache operations on the Oracle database, including passthrough statements, flushing, manual loading, manual refreshing, synchronous writethrough, propagating and dynamic loading.
rpad *varname desiredlength paddingstring*	The RPAD command acts like the SQL function RPAD() with some limitations: ■ The desired length is in bytes, not characters. ■ The padding string is not expanded for string literal escapes, such as unicode escapes. ■ The padding string can contain partial unicode characters or full unicode characters and it may split the padding string in the middle of a multibyte character or surrogate pair. Only variables that are character based (CHAR, VARCHAR) can be padded with the RPAD command.

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 you specify @ with a relative path, the path is relative to the startup directory of ttIsql. When you specify @@, the path 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 3-79 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 set/show 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 3-72.
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.
setvariable *variable_name* := *value*	Sets the value of a scalar bind variable or an element of an array bind variable. For example: setvariable myvar := 'TimesTen'; There must be a space on either side of the assignment operator (:=). For more information, see "Declaring and setting bind variables" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Command	Description
`show {all | attribute}`	Displays the value for the specified `set/show` attribute or displays all the attributes. For a description of accepted attributes, see "Set/show attributes" on page 3-72.
`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.
`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 *filename*, the file name 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). If you estimate statistics with an empty table list, statistics on system tables are updated also, if you have privileges to update the system tables.
`statsupdate [[owner_name_pattern.] table_name_pattern]`	Updates statistics for specified table (or all tables if no table is specified). If `tblName` is an empty string, statistics are estimated for all the current user's tables in the database.

Command	Description
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 to an Oracle database 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 to an Oracle database is enabled, lists tables matching the pattern in the Oracle database. See the alltables command.
tablesize [[*owner_name_pattern*.] *table_name*_pattern]]	For each table that matches the pattern, lists the contents of the ALL_TAB_SIZES view. See the ttComputeTabSizes built-in procedure.
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.
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_name* [*data_type*] [:= *value*]] The syntax for binding multiple values to an array using the variable command is as follows: variable *array_name* '[' *array_size* ']' *data_type*(n):= '[' value1, ... valuex ']'	Declares a bind variable that can be referenced in a statement or displays the definition of the variable if the type is missing. Type can be one of the following: (n), NUMBER, CHAR(n), NCHAR(n), VARCHAR2(n), NVARCHAR2(n), BLOB, CLOB, NCLOB, or REFCURSOR. If only (n) is supplied, it is assumed to be VARCHAR2 (n). Assigns a value to a single variable or multiple values if the data type is an array. You can assign a value later with the setvariable command. For more information, see "Declaring and setting bind variables" in the *Oracle TimesTen In-Memory Database Operations Guide*.
version	Reports version information.

Command	Description
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 to an Oracle database is enabled, lists views matching the pattern in the Oracle database. See the allviews command.
waitfor *expected_result timeoutseconds sqlstatement*	Runs the given statement once a second until the query returns the expected result or a timeout occurs. The query must have only one column and must return exactly one row. Any errors in the query terminate the loop.
waitforresult *expected_result timeoutseconds searchrow searchcol sqlstatement*	Similar to the waitfor command, except that the result can have 1 or more columns. Also, the result can return 0 rows. Runs the given statement once a second until the query returns the expected result or a timeout occurs. The *searchrow* and *searchcol* arguments indicate the ordinal position (1..N) of which row or column should be considered. Use '*' in *searchrow* or *searchcol* to indicate any row or column of the result set could have the expected value. See the waitfor command.
whenever sqlerror	Provide direction on how to handle errors when in ttIsql. For more details, see "Syntax for the WHENEVER SQLERROR command" on page 3-70.
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. Also see "ttXlaDeleteBookmark" in *Oracle TimesTen In-Memory Database C Developer's Guide*. Requires ADMIN privilege or object ownership.

Syntax for the IF-THEN-ELSE command construct

This section provides the syntax for the IF-THEN-ELSE construct. For more details on using the IF-THEN-ELSE command construct, see "Conditional control with the IF-THEN-ELSE command construct" in the *Oracle TimesTen In-Memory Database Operations Guide*.

```
IF [NOT]
   { Literal1 | :BindVariable1 }
   { =  | IN }
   { Literal2 | :BindVariable2 | SelectStatement }
 THEN "ThenCommands"
 [ ELSE "ElseCommands" ] ;
```

The ttIsql IF-THEN-ELSE command has the parameters:

Parameter	Description
IF	The IF command must end in a semicolon (;). The IF command fails if improper syntax is given, the BindVariables do not exist or the SELECT statement fails to execute or does not return just a single column.
NOT	Using NOT reverses the desired result of the condition.
Literal1, *Literal2*	A value that can be part of a comparison.

Parameter	Description
BindVariable1, *BindVariable2*	A bind variable is equivalent to a parameter. You can use the *:BindVariable1* notation for passing bind variables into this construct. The variable can be created and set using the `variable` or `setvariable` `ttIsql` commands.
= | IN	You can use the `IN` operator only with the *SelectStatement*. You can use the `IN` operator with zero or more returned rows. You can use the equal (=) operator only with a single returned row.
SelectStatement	A provided `SELECT` statement must start with `SELECT`. The `SELECT` statement can return only one column. In addition, it can return only one row when the equal (=) operator is provided. The *SelectStatement* is not available if you are not connected to the database.
ThenCommands, *ElseCommands*	All commands in the `THEN` or `ELSE` clauses must be delimited by a semicolon and cannot contain embedded double quotes. These clauses can conditionally execute `ttIsql` commands, such as `host` or `run`, which cannot be executed through PL/SQL. You can use the `CALL` statement within the `THEN` or `ELSE` clauses. You cannot use PL/SQL blocks.

Restrictions for the `IF-THEN-ELSE` construct are as follows:

- You cannot compare variables of the LOB data type.
- The values are compared case-sensitive with `strcmp`. A character padded value might not match a `VARCHAR2` because of the padding.

Syntax for the WHENEVER SQLERROR command

Execute the `WHENEVER SQLERROR` command to prescribe what to do when a SQL error occurs. For more details and examples on how to use the `WHENEVER SQLERROR` command, see "Error recovery with the WHENEVER SQLERROR" command in the *Oracle TimesTen In-Memory Database Operations Guide*.

```
WHENEVER SQLERROR { ExitClause | ContinueClause | SUPPRESS |
     SLEEP Number | ExecuteClause }
```

When you specify `EXIT`, always exit `ttIsql` if an error occurs. *ExitClause* is as follows:

```
EXIT [ SUCCESS | FAILURE | WARNING | Number | :BindVariable ]
 [ COMMIT | COMMIT ALL | ROLLBACK ]
```

When you specify `CONTINUE`, `ttIsql` continues to the next command, even if an error occurs. *ContinueClause* is as follows:

```
CONTINUE [ COMMIT | COMMIT ALL | ROLLBACK | NONE ]
```

Execute specified commands before continuing. *ExecuteClause* is as follows:

```
EXECUTE "Cmd1;Cmd2;...;"
```

The `WHENEVER SQLERROR` command options are as follows:

- `EXIT`: Always exit `ttIsql` if an error occurs. Specify what is performed before `ttIsql` exits with one of the following. `SUCCESS` is the default option for `EXIT`.
  - `SUCCESS` or `FAILURE` or `WARNING`: Return `SUCCESS` (value 0), `FAILURE` (value 1), or `WARNING` (value 2) to the operating system after `ttIsql` exits for any SQL error.

  - *Number*: Specify a number from 0 to 255 that is returned to the operating system as a return code. Once ttIsql exits, you can retrieve the error return code with the appropriate operating system commands. For example, use echo $status in the C shell (csh) or echo $? in the Bourne shell (sh) to display the return code.

    The return code can be retrieved and processed within batch command files to programmatically detect and respond to unexpected events.

  - *:BindVariable*: Returns the value in a bind variable that was previously created in ttIsql with the variable command. The value of the variable at the time of the error is returned to the operating system in the same manner as the *Number* option.

    Note: The bind variable used within the WHENEVER SQLERROR command cannot be defined as a LOB, REFCURSOR, or any array data type.

  In addition, you can specify whether to commit or rollback all changes before exiting ttIsql.

  - COMMIT: Executes a COMMIT and saves changes only in the current connection before exiting. The other connections exit with the normal disconnect processing, which rolls back any uncommitted changes.
  - COMMIT ALL: Executes a COMMIT and saves changes in all connections before exiting.
  - ROLLBACK: Before exiting, executes a ROLLBACK and abandons changes in the current connection and, by default, in all other connections. The other connections exit with the normal disconnect processing, which automatically rolls back any uncommitted changes.

- CONTINUE: Do not exit if an error occurs. The SQL error is displayed, but the error does not cause ttIsql to exit. The following options enable you to specify what is done before continuing to the next ttIsql command:
  - NONE: This is the default. Take no action before continuing.
  - COMMIT: Executes a COMMIT and saves changes in the current connection before continuing.
  - COMMIT ALL: Executes a COMMIT and saves changes in all connections before continuing.
  - ROLLBACK: Before continuing, executes a ROLLBACK and abandons changes in the current connection and, by default, in all other connections. The other connections exit with the normal disconnect processing, which automatically rolls back any uncommitted changes.
- SUPPRESS: Do not show any error messages and continue.
- SLEEP: Sleep for a specified number of seconds before continuing.
- EXECUTE: Execute specified commands before continuing. Each command is separated from the other commands by a semicolon (;). If any command triggers additional errors, those errors may cause additional actions that could potentially result in a looping condition.

Set/show attributes

Also see the list of `ttIsql` "Commands" on page 3-57. Some commands appear here as attributes of the `set` command. In that case, you can use them with or without the `set` command.

Boolean attributes can accept the values "`ON`" and "`OFF`" or "`1`" and "`0`".

The `ttIsql set` command has the 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.
`autovariables [1|0]`	Turns `autovariables` off and on. TimesTen creates an automatic bind variable with the same name as each column in the last fetched row. You can use an automatic bind variable in the same manner of any bind variable. For more information, see "Automatically creating bind variables for retrieved columns" in the *Oracle TimesTen In-Memory Database Operations Guide*.
`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 3-82 for an explanation of the default.
`dynamicloadenable [on|off]`	Enables or disables dynamic load of data from an Oracle database to a TimesTen dynamic cache group. By default, dynamic load of data from an Oracle database 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.

Attribute	Description
err | error |errors [.*objecttype*[*schema*.] *name*]	With the show command, displays 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.
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.
loboffset *n*	Specifies the offset into the LOB that ttIsql should use as the starting point when it prints the resulting value of a LOB. For example if the value of the LOB is ABCEDFG, and the offset is 4, ttIsql prints DEFG, skipping the first 3 bytes. The behavior is the same as LOBOFFSET in SQL*Plus.
long *n*	Reports or controls the maximum number of characters for CLOB or BLOB data or the maximum number of bytes for BLOB data that are displayed when fetched or printed. The default value is 80. The command setting is valid for all connections in a session.
longchunksize *n*	Specifies the size of the chunk that ttIsql uses to get LOB data.
multipleconnections [1 | ON] mc [1 | ON]	Reports or enables handling of multiple connections.By default, ttIsql enables 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.

Attribute	Description
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.
passthrough [0|1|2|3|4|5]	Sets the TimesTen 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 the Oracle database. 4 - All SELECT statements on global cache groups tables that cannot use dynamic load are executed on the Oracle database. 5 - All SELECT statements on global cache groups tables that cannot use dynamic load are executed on the Oracle database. The SELECT statement is not executed until after all committed changes to the global cache group are propagated to the Oracle database. 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. When you set the prefetch count to 0, TimesTen uses a default prefetch count. The default prefetch value is isolation level specific. In read committed isolation mode, the default value is 5. In serializable isolation mode, the default value is 128. You can also enable this attribute without specifying the set command.

Attribute	Description
`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.
`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.
`showcurrenttime [1|true|on] | [0|false|off]`	Enable or disable printing of the current wall clock time.
`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 TimesTen Cache operations on the Oracle database, including passthrough statements, flushing, manual loading, manual refreshing, synchronous writethrough, propagating, and dynamic 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.

Attribute	Description
tryhash [1|0]	Enables or disables use of hash indexes by the optimizer at the transaction level. AutoCommit must be off. You can also enable this attribute without specifying the set command.
trymaterialize [1|0]	Enables or disables materialization by the optimizer at the transaction level. 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 the optimizer at the transaction level. AutoCommit must be off. You can also enable this attribute without specifying the set command.
trynestedloopjoin [1|0]	Enables or disables use of nested loop joins by the optimizer at the transaction level. AutoCommit must be off. You can also enable this attribute without specifying the set command.
tryrowid [1|0]	Enables or disables rowID scan hint by the optimizer at the transaction level.
tryrowlocks [1|0]	Enables or disables use of row-level locking by the optimizer at the transaction level. 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 the optimizer at the transaction level. 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 the optimizer at the transaction level. 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 at the transaction level. 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 the optimizer at the transaction level. AutoCommit must be off. You can also enable this attribute without specifying the set command.
trytmprange [1|0]	Enables or disables use of temporary range indexes by the optimizer at the transaction level. AutoCommit must be off. You can also enable this attribute without specifying the set command.
tryrange [1|0]	Enables or disables use of range indexes by the optimizer at the transaction level. AutoCommit must be off. You can also enable this attribute without specifying the set command.

Attribute	Description
`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, `ttIsql` ignores the remainder of the line.

'`--`' 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 `ttIsql` keystroke shortcuts are:

Keystroke	Action
Left Arrow	Moves the insertion point left (back).
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 that 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` `set`/`show` 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/tt1122/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
    Range: 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
    UserBoyerMooreStringSearch: 0
    DynamicLoadEnable: 1
    DynamicLoadErrorMode: 0
    NoRemRowIdOpt: 0

Current Join Order:
    <>

Command
```

Prepare and execute an SQL statement.

```
ttIsql (c) 1996-2011, 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.
```

To run a `SELECT` query until the result "`X`" is returned or until the query times out at 10 seconds, 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> waitfor X 10 select * from dual;
Command>
```

Example of managing XLA bookmarks

You can use the `xlabookmarkdelete` command to both check the status of the current XLA bookmarks and delete them. This command requires `XLA` privilege or object ownership.

For example, when running the XLA application, `'xlaSimple'`, you can check the bookmark status by entering:

```
Command> xlabookmarkdelete;

XLA Bookmark: xlaSimple
  Read Log File: 0
  Read Offset: 630000
  Purge Log File: 0
  Purge Offset: 629960
  PID: 2808
  In Use: No
1 bookmark found.
```

To delete the bookmark, enter:

```
Command> xlabookmarkdelete xlaSimple;
Command>
```

Example parameters using "variable" and "print"

Substitution in `ttIsql` is modeled after substitution in SQL*Plus. To enable the substitution feature, use `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 with only one substitution character specified 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

The `ttIsql` utility supports only generic `REF CURSOR` variables, not specific `REF CURSOR` types.

Multiple `ttIsql` commands are allowed per line separated by semicolons. If a named identifier includes a `#` character in the name, then to place several SQL statements on a single line, you must quote the statements that include the named identifier. For example:

```
% ttisql -e 'drop table "t#" ; create table "t#" ( c1 int );' -dsn mydb
```

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 stop the parameter entry process.

```
Enter Parameter 1> N'XY';
```

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 data file.
- Restores the migrate object from the binary data file into a TimesTen database.
- Examines the contents of a binary data file created by this utility.

Migrated objects include:

- Tables
- Cache group definitions
- Views and materialized views
- Materialized view log definitions
- Sequences
- Replication schemes

Use the `ttMigrate` utility when upgrading major release versions of TimesTen, since database checkpoint and log files are not compatible between major releases. 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, TimesTen does not migrate users and user privileges. When you migrate a database between releases of Release 11.2.1 or into a release later than Release 11.2.1, TimesTen migrates users and user privileges.

Binary files produced by this utility are platform-dependent. For example a binary file produced on Windows must be restored on Windows. In client/server mode, use `ttMigrateCS` (UNIX only) utility to copy data between platforms.

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.

Binary files produced by this utility are platform-specific. For example, a binary file produced on Windows 64-bit must be restored on Windows 64-bit. To copy data between platforms or bit levels, use `ttMigrate` with the `ttMigrateCS` client/server version (or Windows equivalent). On Windows systems, you can do the equivalent by using `ttMigrate` to connect to the source system from the target system through a defined TimesTen client DSN.

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 TimesTen Cache, then CACHE_MANAGER is also required.

Using the -c option to capture an entire database requires the ADMIN privilege. If the database is involved in replication or TimesTen Cache, then CACHE_MANAGER is also required. 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 data file, use:

```
ttMigrate {-a | -c} [-v verbosity] [-nf] [-nr] [-fixNaN] [-saveAsCharset charset]
[-relaxedUpgrade | -exactUpgrade]
[-convertTypesToOra | -convertTypestoTT]]
[-activeDML | -noActiveDML]
{-connStr connection_string | DSN} data file [objectOwner.]objectName
```

To restore a database from a binary data file created by this utility, use:

```
ttmigrate -r  [-C ckptFreq] [-v level] [-nf] [-nr] [-fixNaN] [-numThreads n]
[-updateStats | -estimateStats percent] [-relaxedUpgrade | -exactUpgrade]
[-inline rule] [-noCharsetConversion] [-cacheUid uid [-cachePwd pwd]]
[-autorefreshPaused] [-convertTypesToOra | -convertTypesToTT]
[-restorePublicPrivs] [-localhost host]
[-delayFkeys | -noDelayFKeys]
{DSN | -connstr connStr} dataFile [objectOwner.objectName...]
```

To list or display the contents of a binary data file 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 3-91 for more details.

Option	Description
-activeDML |-noActiveDML	Saves all tables in a foreign key hierarchy in a single transaction, maintaining consistency between these tables when there is active DML during the ttMigrate -c operation. If -noActiveDML is specified, ttMigrate saves each table in its own transaction, regardless of whether it is the parent or the child of a foreign key. Use this option if there is no active DML during the ttMigrate -c operation. -noActiveDML is the default.
-c	Create mode: Creates an original binary data file. See "Create mode (-c) and Append mode (-a)" on page 3-91 for more details.
-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 TimesTend Database data types to Oracle Database data types or Oracle Database data types to TimesTen data types. These options require the -relaxedUpgrade option. In TimesTen 11.2.2 the default type mode is ORACLE type mode. The -convertTypesToOra is useful when migrating older databases into TimesTen 11.2.2. 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 Database data type conversions" on page 3-95 and "Oracle Database to TimesTen data type conversions" on page 3-96.
-d	Selects Describe mode. Displays a short description of the objects in the data file. See "Describe mode (-d)" on page 3-94 for more details.
-D	Selects Long-describe mode. Displays a full description of the objects in the data file. See "Long-describe mode (-D)" on page 3-95 for more details.
dataFile	The path name of the data file 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.

Option	Description
`-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. `ttMigrate` ignores this option 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.

`-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 `-exactUpgrade` 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`. `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 data file. See "List mode (-l) and Long-list mode (-L)" on page 3-94 for more details.
`-L`	Selects Long-list mode. Lists the names of database objects in the specified data file and other details about the database objects. See "List mode (-l) and Long-list mode (-L)" on page 3-94 for more details.
`-r`	Selects Restore mode. Restores a database from a binary data file created by this utility. See "Restore mode (-r)" on page 3-93 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.

Option	Description
-relaxedUpgrade	Save or restore the tables in a way that is compatible with a replication scheme that uses TABLE DEFINITION CHECKING RELAXED. Use of this option may cause the restored tables to be slightly more compact and slightly faster to access than otherwise. ttMigrate ignores this option when the -a option is given. This option should not be used in combination with a replication scheme that uses TABLE DEFINITION CHECKING EXACT, or else replication may no longer work. The default is -exactUpgrade.
-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.
-exactUpgrade	Save or restore the tables in a way that is compatible with a replication scheme that uses TABLE DEFINITION CHECKING EXACT ttMigrate ignores this option when the -c or -a options are given. This option should not be used in combination with a replication scheme that uses TABLE DEFINITION CHECKING RELAXED, or else replication may no longer work. INLINE variable-length columns cannot successfully be replicated to NOT INLINE columns. This is the default.
-saveAsCharset *charset*	Saves an object in the specified connection character set. ttMigrate returns an informational message if the connection character set is different from the database character set. If this option is not set, by default, ttMigrate saves the migrated object in the database character set.
-updateStats	Specifies that ttMigrate should update statistics on restored tables and materialized views. ttMigrate ignores this option 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 (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. ttMigrate ignores the -v option in List, Long-list, Describe and Long-describe modes.
-V | -version	Prints the release number of ttMigrate and exits.

The following ttMigrate options are available in restore mode (-r) only:

Option	Description
-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 database tables to TimesTen cached tables using: ■ The types of the columns in the Oracle database tables. ■ The types of the columns stored in the migration file. ■ The TimesTen-to-Oracle type mapping rules. 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. For more information, see "Mappings between Oracle Database and TimesTen data types" in *Oracle TimesTen Application-Tier Database Cache User's Guide*.
-delayFkeys | -noDelayFkeys	-delayFkeys - Delay creation of foreign keys until all tables have been restored. This can improve performance for parallel migration (ttMigrate -numThreads). -noDelayFkeys - Create foreign keys as part of the CREATE TABLE operation. -noDelayFkeys is the default.
-localhost *hostName*	Explicitly identifies the name or IP address of the local host when restoring replicated tables.
-noCharsetConversion	Restores data, retaining the connection character set that is stored in the data file. ttMigrate does not convert the connection character set to match the database character set. If not set, ttMigrate restores the data and converts the connection character set to be the same as the database character set. See also: -saveAsCharset. This option may be useful for legacy TimesTen users who may have migrated pre-11.2.2 data into a 11.2.2 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 want 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. By default, the ttMigrate utility does not restore privileges granted to PUBLIC. You must explicitly specify this option to restore privileges to PUBLIC.

Modes

Create mode (-c) and Append mode (-a)

In create mode, ttMigrate saves migrate objects from a TimesTen database into a new binary data file. If the data file does not exist, ttMigrate creates it. Otherwise, ttMigrate overwrites the existing file, destroying its contents.

The data file 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 data file. If the data file 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. After restoring a cache group with ttMigrate -r, reset its AUTOREFRESH STATE to ON by using the ALTER CACHE GROUP statement (this can be done programmatically or with the ttIsql utility.

- 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 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 data file, 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 data file, 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 data file 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 `-exactUpgrade` option is set during restore.

By default, ttMigrate restores all tables and cache groups in the data file. 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 data file, including cached tables and the replication scheme TTREP tables.

In Long-list mode, ttMigrate lists the names of database objects in the data file, 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 Database data type conversions

Both TimesTen and Oracle Database data types are supported in TimesTen 11.2.2 When migrating a database from an earlier version of TimesTen to TimesTen release 11.2.2, 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 prior to 11.2.2 do not support Oracle Database 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 Database 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`

From TimesTen Type	To Oracle Type
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 Database to TimesTen data type conversions

When migrating tables backward from TimesTen release 11.2.2 to an earlier version of TimesTen, you may need to convert Oracle Database data types to TimesTen data types, as the Oracle database data types were not supported in releases before 11.2.2.

To convert from Oracle Database 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

Pre-7.0 TimesTen Type	Oracle Type	Converted Type
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 *Oracle TimesTen In-Memory Database SQL Reference* and "Mappings between Oracle Database and TimesTen data types" in *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Return codes

The ttMigrate utility restore (-r) and create (-c) commands return the following 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 SalesDS sales.ttm
```

This command appends all database objects in the SalesDS database owned by user MARY to sales.ttm:

```
ttMigrate -a SalesDS sales.ttm MARY.%
```

This command restores all database objects from sales.ttm into the SalesDS database:

```
ttMigrate -r SalesDS sales.ttm
```

This command restores MARY.PENDING and MARY.COMPLETED from sales.ttm into SalesDS (migrate objects are case-insensitive):

```
ttMigrate -r 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 3-96 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 database tables. Cache group migration fails when there is a mismatch in the length or length semantics of any of its cached tables.

The connection attribute PassThrough with a nonzero value is not supported with this utility and returns an error.

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 `-relaxedUpgrade` option. Using the default `-exactUpgrade` option results in an error. The `INLINE` column attributes are 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, ensure 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` together with `ttMigrateCS` (client server version of `ttMigrate`) to migrate databases between 32- and 64-bit platforms or bit levels. You must use the `-relaxedUpgrade` option when restoring data on a new bit-level. In the case of changing bit-levels, the database cannot be involved in a replication scheme. Follow the examples in "Moving a database between 32-bit and 64-bit platforms" in the *Oracle TimesTen In-Memory Database Installation Guide*.

- 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.

It is recommended that you do not run DDL SQL commands while running `ttMigrate` to avoid lock contention issues for your application.

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 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. You can use this option to assign the port for the TimesTen cluster agent. See "Using Oracle Clusterware to Manage Active Standby Pairs" in *Oracle TimesTen In-Memory Database 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

You must shut down all TimesTen operations 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
      [-localIP localIPAddress] [-remoteIP remoteIPAddress]
      [-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][-nThreads]]
      [-keepCG [-cacheUid cacheUid [-cachePwd cachePwd]]
         [-recoveringNode | -deferCacheUpdate]
      | -nokeepCG]
      [-remoteDaemonPort portNo] [-verbosity {0|1|2}]
      [-localhost localHostName]
      {destDSN | -connStr connectionString}
```

ttRepAdmin operations

Use the `ttRepAdmin` utility 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*	Host name or TCP/IP address of the subscriber host.
-state start	Sets the state of replication for the subscriber. start (default) - Starts replication to the subscriber. stop - Stops replication to the subscriber, discarding updates. pause - Pauses the replication agent, preserving updates. See "Setting the replication state of subscribers" in *Oracle TimesTen In-Memory Database Replication Guide* for more information.
-state stop	
-state pause	
-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 an internal user with the ADMIN privilege on the source database.
- You must run ttRepAdmin on the target host.
- The DSN specified must be a direct-mode DSN, not a server DSN.

Before running the ttRepAdmin -duplicate command, use ttStatus to ensure the replication agent is started for the source database.

```
ttRepAdmin -duplicate -from srcDataStoreName
       -host srcDataStoreHost
      [-localIP localIPAddress] [-remoteIP remoteIPAddress]
      [-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] [-nThreads]]
      [-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. ttRepAdmin ignores the option 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 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 *srcDataStoreHost*	Defines the host name 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. If you cannot connect to the Oracle database or the Oracle database is down, then specify the -recoveringNode option when the -duplicate is being used to recover a failed node for a replication scheme that includes all AWT or incremental autorefresh cache groups. Otherwise, specify the -deferCacheUpdate option. These options defer changes to metadata on the Oracle database (that is used to manage AWT or incremental autorefresh cache groups) until after the duplicate operation completes, the cache and replication agents are started, and these agents can connect to the Oracle database. See "Duplicating a database" in the *Oracle TimesTen In-Memory Database Replication Guide* for more information.
-localhost *hostName*	Use with -duplicate and -setMasterRepStart to explicitly identify the name or IP address of the local host.
-localIP *localIPAddress*	Specifies the alias or IP (IPv4 or IPv6) address of the local network interface to be used. If not specified, ttRepAdmin chooses any compatible interface.
-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.
-nThreads *n*	Used with the -initCacheDr option, -nThreads indicates the number of threads used to truncate the Oracle database tables and push the data in the cache into Oracle during the initialization process.
-PWD *pwd*	The password of the internal user specified in the -UID option.
-PWDCrypt *encryptedPwd*	The encrypted password of the user specified in the -UID option.

Option	Description
-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, RAM Policy for the database is set to manual. Use the ttAdmin utility to make further changes to the RAM policy.
-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.
-remoteIP *remoteIPAddress*	Specifies the alias or IP (IPv4 or IPv6) address of the remote or destination network interface to be used. If not specified, ttRepAdmin chooses any compatible interface.
-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. ttRepAdmin ignores the option 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. This must be an internal user.
-verbosity {0 | 1 | 2}	Provide details of the communication steps within the duplicate process and reports progress information about the duplicate transfer. 0 (default) - No diagnostics are returned. 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 operating system 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 host name 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 `AwtMonitorConfig` 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 the configuration of replicated databases" in *Oracle TimesTen In-Memory Database Replication Guide* for more information.
`-showstatus`	Reports the current status of the specified replicated database. See "Use ttRepAdmin to show replication status" in *Oracle TimesTen In-Memory Database 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 *Oracle TimesTen In-Memory Database 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 the Oracle database.
- 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 the Oracle database.
- Time since monitoring was started.
- Total number of TimesTen row operations: The total number of rows updated in AWT cache groups since monitoring was enabled.
- Total number of TimesTen transactions: The total number of transactions in AWT cache groups since monitoring was enabled.
- Total number of flushes to Oracle: The total number of times that TimesTen data has been sent to the Oracle database.

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 want 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 Replication Guide*.
For upgrade examples, see "TimesTen 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. The `LogDir` attribute enables 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 data store attributes, except for the `LogDir` attribute.

For an overview of the TimesTen backup and restore facility, see "Migration, Backup, and Restoration" in the *Oracle TimesTen In-Memory Database Installation 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`.

Option	Description
-noconn	To ensure that the restore was successful, ttRestore connects to the database as a last step. This option disables that last connect. *We recommend that you specify this option for best performance*. If this option is not specified, the database is loaded into memory and unloaded from memory.
-V | -version	Prints the release number of ttRestore and exits.

Examples

```
ttRestore -dir /users/pat/TimesTen/backups
-fname FastInsBkup 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 origDSN
ttRestore -dir /users/rob/tmp -fname restored restoredDSN
```

The value of fname is the name that you want for the prefix portion of the backup file name.

On UNIX, to restore a tape backup to the FastIns database, use:

```
dd bs=64k if=/dev/rmt0 | ttRestore -i FastIns
```

Notes

The ttBackup utility and the ttRestore utility backup and restore databases only when the first three numbers of the TimesTen release and the platform are the same. For example, you can backup and restore files between releases 11.2.2.2.0 and 11.2.2.3.0. You cannot backup and restore files between releases 11.2.1.9.0 and 11.2.2.3.0. You can use the ttBulkcp or ttMigrateCS (UNIX only) utility to migrate databases across major releases or operating systems. You can use ttMigrate together with ttMigrateCS (client server version of ttMigrate) to migrate databases between 32- and 64-bit platforms or bit levels. You must use the -relaxedUpgrade option when restoring data on a new bit-level. In the case of changing bit-levels, the database cannot be involved in a replication scheme. Follow the examples in "Moving a database between 32-bit and 64-bit platforms" in the *Oracle TimesTen In-Memory Database Installation Guide*.

You can backup databases containing cache groups with the ttBackup utility. However, when restoring such a backup, special consideration is required as the restored data within the cache groups may be out of date or out of sync with the data in the back end Oracle database. See the section on "Backing up and restoring a database with cache groups" in the *Oracle TimesTen Application-Tier Database Cache User's Guide* for details.

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 `ttIsql` utility can play back the output of `ttSchema` utility to rebuild the full schema of a database.

On UNIX, this utility is supported for TimesTen Data Manager DSNs. For TimesTen Client DSNs, use the utility `ttSchemaCS`.

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 |synonyms | plsql} [,...] ]
         [-plsqlAttrs | -noplsqlAttrs]
         [-plsqlCreate |-[no]plsqlCreateOrReplace]
         {-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.
-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 the Oracle database. The -c option 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 |synonyms | 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 | -[no]plsqlCreateOrReplace	If -plsqlCreate is specified, ttSchema emits CREATE PROCEDURE, CREATE PACKAGE or CREATE FUNCTION statements for PL/SQL program units. If -plsqlCreateOrReplace (default) is specified, ttSchema emits CREATE or REPLACE statements.
-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 specific objects

Return only the materialized views and sequences for the `orderdsn` database.

```
% ttSchema -list views,sequences orderdsn
-- Database is in Oracle type mode
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 3–6 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. The `ttSchema` utility prints out the

partition numbers for the columns that are not in the initial partition. The initial partition is `0`, so partition 1 as printed by `ttSchema` is secondary partition `1`, not the initial partition. For more details on partitions, see "Understanding partitions when using ALTER TABLE," in the "ALTER TABLE" section of the *Oracle TimesTen In-Memory Database SQL Reference*.

The connection attribute `PassThrough` with a nonzero value is not supported with this utility and returns an error.

Output is not guaranteed to be compatible with DDL recognized by previous releases of TimesTen.

It is recommended that you do not run DDL SQL commands while running `ttSchema` to avoid lock contention issues for your application.

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. You can use this utility on existing tables or to estimate table sizes when creating tables. If you do not specify an owner, `ttSize` prints size information for all tables of the given table 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 memory required for LOB 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. When no rows are being inserted into the table, computations do not include LOB columns.

The table is scanned when this utility is called. Avoid the scan of the table by specifying an optional non-`NULL` *frac* value, which should be between `0` and `1`. The `ttSize` utility uses this value 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.

Option	Description
-frac *frac*	Specifies the estimated average fraction of out-of-line VARCHAR or VARBINARY column sizes that will be used. If this option is omitted and the table contains out-of-line variable sized columns, a table scan is done to determine the average sizes. If the table is empty, the fraction is estimated to be 0.5 (50%) filled.
-h -help -?	Prints a usage message and exits.
-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.
-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 ttSize utility uses the number of rows in the existing table to estimate the table space, or uses 1 row if the table is empty.
-V | -version	Prints the release number of ttSize and exits.

Examples

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 -tbl Nameid -rows 200000 FixedDs

Rows = 200000

Total in-line row bytes = 7139428

Total = 7139428
```

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 Views Reference*.)

LOB columns are treated similar to var-type columns, unless there are no rows being inserted into the table. The average size computation does not include LOB columns in such cases.

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.

ttStats

Description

The `ttStats` utility monitors database metrics (statistics, states, and other information) or takes and compares snapshots of metrics. It can perform the following functions.

- Monitor and display database performance metrics in real-time, calculating rates of change during each preceding interval.
- Collect and store snapshots of metrics to the database then produce reports with values and rates of change from a specified pair of snapshots. (These functions are performed through calls to the `TT_STATS` PL/SQL package.)

TimesTen gathers metrics from TimesTen system tables, views, and built-in procedures. In reports, this includes information such as a summary of memory usage, connections, and load profile, followed by metrics (as applicable) for SQL statements, transactions, PL/SQL memory, replication, logs and log holds, checkpoints, cache groups, cache grid, latches, locks, XLA, and TimesTen connection attributes. Monitoring displays a smaller set of key data, as shown later in this section.

For client DSNs, use the `ttStatsCS` version of the utility (UNIX or Windows).

There are three modes of operation:

- Monitor mode (default mode): Tracks database performance in real-time by monitoring a pre-determined set of metrics, displays those metrics (primarily those whose values have changed since the last display), and calculates rates of change in the values where appropriate. Information is output to the standard output for display to the user and is not stored to disk.

  If the duration or number of iterations is not specified, the monitoring runs until interrupted with `Ctrl-C`.

  > **Note:** The set of metrics displayed in monitor mode is subject to change, depending on changes to the system tables and built-in procedures from which metrics are gathered.

- Snapshot mode: Takes a snapshot of metrics, according to the capture level, and stores them to database `SYS.SNAPSHOT_`*`XXXX`* system tables. Once the snapshot is taken, its ID number is displayed to the standard output. The capture level applies only to metrics from `SYS.SYSTEMSTATS`. For metrics from other sources, the same data are collected regardless of the capture level.

  By default, a "typical" set of metrics is collected, which suits most purposes, but you can specify a reduced "basic" set of metrics, all available metrics, or only those metrics from sources other than `SYSTEMSTATS`.

- Report mode: Generates a report from two specified snapshots of metrics. Reports are in HTML format by default, but you can request plain text format. You can specify an output file or display output to the standard output. For those familiar with Oracle Database performance analysis tools, the `ttStats` reports are similar in nature to Oracle Automatic Workload Repository (AWR) reports.

In monitor mode, the overhead of reading from the database is avoided. In snapshot mode and report mode, the `ttStats` utility is a convenient front end to the `TT_STATS` PL/SQL package provided by TimesTen. Refer to "TT_STATS" in *Oracle TimesTen In-Memory Database PL/SQL Packages Reference* for details on that package.

Notes: The `ttStats` utility has the following dependencies and limitations:

- Monitor mode requires features added to the `SYS.SYSTEMSTATS` table in TimesTen release 11.2.2.4.0.
- Snapshot and report modes require the `TT_STATS` PL/SQL package, added in TimesTen release 11.2.2.5.0.
- The utility cannot be used if you are connecting to TimesTen through a driver manager.

Snapshots are stored in a several TimesTen `SYS.SNAPSHOT_`*xxxxx* system tables. (For reference, these tables are listed in "Tables and views reserved for internal or future use" in *Oracle TimesTen In-Memory Database System Tables and Views Reference*.)

For information about built-in procedures mentioned, and the data they gather, see Chapter 2, "Built-In Procedures".

Required privilege

- **Monitor mode**: No special privilege is required to run monitor mode, but `ADMIN` privilege is required for the monitoring information to include data from the `ttSQLCmdCacheInfo` built-in procedure and `transaction_log_api` (XLA) table.
- **Snapshot and report mode**: By default, only the instance administrator has privilege to run in snapshot or report mode, due to security restrictions of the `TT_STATS` PL/SQL package. Any other user, including an `ADMIN` user, must be granted `EXECUTE` privilege for the `TT_STATS` package by the instance administrator or by an `ADMIN` user, such as in the following example:

```
GRANT EXECUTE ON SYS.TT_STATS TO scott;
```

Syntax

```
ttStats [-h | -help]
ttStats [-V | -version]
ttStats [-monitor] [-interval seconds]
        [-duration seconds] [-iterations count]
           {DSN | -connStr connectionString}
ttStats  -snapshot [-level capture_level] [-notes snap_desc]
           {DSN | -connStr connectionString}
ttStats  -report [-snap1 snapid1 -snap2 snapid2]
        [-html | -text] [-outputFile filename]
           {DSN | -connStr connectionString}
```

Note: Specify only one of `-monitor`, `-snapshot`, or `-report`.

Options

`ttStats` has the options:

Option	Description
-h	Prints the list of options and exits.
-help	**Note**: This is also the result if nothing is entered on the ttStats command line, or if options are entered without a DSN or connection string.
-V	Prints the TimesTen release number and exits.
-version	
-monitor	Run in real-time monitor mode. Monitors a pre-determined set of metrics and repeatedly displays the metrics and rates of change. Unlike in snapshot mode, nothing is stored to the database. **Note**: This is the default mode if neither -monitor, -snapshot, nor -report is specified.
-interval *seconds*	For monitor mode, this is the time interval between sets of metrics that are displayed, in seconds. The default is 10 seconds. Shorter intervals may negatively impact system performance.
-duration *seconds*	For monitor mode, this is the duration of how long ttStats runs, in seconds. After this duration, the utility exits. Also see information for the -iterations option.
-iterations *count*	For monitor mode, this is the number of iterations ttStats performs in gathering and displaying metrics. After these iterations, the utility exits. **Note**: If you specify both -duration and -iterations, monitoring stops when the first of the two limits is reached. If you specify neither, monitoring continues until interrupted by Ctrl-C
-snapshot	Collect a snapshot of metrics according to the capture level and store the metrics in the database. Once the snapshot is captured, its ID number is displayed. **Notes**: ■ TimesTen gathers all SYSTEMSTATS when you take a snapshot, but only those within the specified capture level have meaningful accumulated values. Metrics outside of the specified level have a value of 0 (zero). ■ This option is implemented by a call to the CAPTURE_SNAPSHOT procedure of the TT_STATS PL/SQL package.
-level *capture_level*	For snapshot mode, this is the level of metrics to capture. The possible settings are as follows: ■ 0: For metrics outside of SYS.SYSTEMSTATS only. ■ 1: For only "basic" metrics. ■ 2 (default): For "typical" metrics. This includes the basic metrics. This level is appropriate for most purposes. ■ 3: For all available metrics. Use the same level for any two snapshots to be used in a report. **Notes**: ■ These levels correspond to the capture levels NONE, BASIC, TYPICAL, and ALL for the TT_STATS PL/SQL package. ■ The capture level applies only to metrics from the SYS.SYSTEMSTATS table. For metrics from other sources, the same data are collected regardless of the capture level.
-notes *snap_desc*	For snapshot mode, optionally use this to provide any description or notes for the snapshot, for example to distinguish it from other snapshots.

Option	Description
`-report`	Generate a report from two specified snapshots, in HTML format by default. Use snapshots taken at the same capture level. **Notes**: ■ If you do not specify any snapshot IDs, a list of available snapshots (with date, time, capture level, and any notes) is displayed and you are prompted to enter each of the desired IDs. If you specify only one snapshot ID, you are told that you must enter two—reenter the command, specifying two snapshots. ■ This option is implemented by a call to the `GENERATE_REPORT_HTML` or (if the `-text` option is used) the `GENERATE_REPORT_TEXT` procedure of the `TT_STATS` PL/SQL package.
`-snap1` *snapid1*	For report mode, this is the snapshot ID of the first snapshot.
`-snap2` *snapid2*	For report mode, this is the snapshot ID of the second snapshot.
`-outputFile` *filename*	For report mode, optionally specify a file path and name where the report is to be written. If no file is specified, TimesTen writes the to the standard output.
`-html` | `-text`	For report mode, specify HTML or plain text output format. **Note**: It is not necessary to specify `-html`. If you specify no format, the report is in HTML format by default.
`-connStr` *connstring* or *DSN*	To specify and connect to the database from which to gather metrics, do one of the following: ■ Specify an ODBC connection string, preceded by `-connStr`. ■ Specify a DSN (data source name), without `-connStr`, at the end of the command line. See "Specifying Data Source Names to identify TimesTen databases" in *Oracle TimesTen In-Memory Database Operations Guide* for information about TimesTen DSNs.

Examples

This section provides examples of `ttStats` monitoring and report output.

> **Note:** Examples are for illustrative purposes only. Details are subject to change.

Monitor example

This section shows sample output from monitor mode.

```
% ttStats sampledb_1122

Connected to TimesTen Version 11.02.02.0005 TimesTen Cache version 11.2.2.5.0.

Waiting for 10 seconds for the next snapshot
Description                          Current  Rate/Sec  Notes
date.2012-Dec-20 16:49:25      -869676175380467200           1  sample #, not rate
connections.count                         12
db.size.temp_high_water_mark.kb         7153       7
lock.locks_granted.immediate             832       1
log.log_bytes_per_transaction              0
loghold.bookmark.log_force_lsn    0/12027904
loghold.bookmark.log_write_lsn    0/12050944
```

```
loghold.checkpoint_hold_lsn       0/12025856                  sampledb_1122.ds0
loghold.checkpoint_hold_lsn       0/12023808                  sampledb_1122.ds1
stmt.executes.count                       44       1
stmt.executes.selects                     32       1
```

Note: The number following the date and time is a numeric representation of the time of the snapshot and can be ignored.

The following command line example specifies that monitoring should stop after two iterations and uses a connection string to set a connection attribute value.

```
% ttStats -iterations 2
-connStr "DSN=sampledb_1122;PLSQL_MEMORY_ADDRESS=20000000"
```

Snapshot example

The following examples take two snapshots at the default typical level:

```
% ttStats -snapshot sampledb_1122

Connected to TimesTen Version 11.02.02.0005 TimesTen Cache version
11.2.2.5.0.
Snapshot 1 at TYPICAL level was successfully captured.

% ttStats -snapshot sampledb_1122

Connected to TimesTen Version 11.02.02.0005 TimesTen Cache version
11.2.2.5.0.
Snapshot 2 at TYPICAL level was successfully captured.
```

Report examples

The following example creates a report from the snapshots generated in the previous section.

```
% ttStats -report -outputFile testreport.html -snap1 1 -snap2 2 sampledb_1122

Connected to TimesTen Version 11.02.02.0005 TimesTen Cache version
11.2.2.5.0.
Report testreport.html was created.
```

The rest of this section shows excerpts from tables of metrics that a `ttStats` report generates. This output was produced using the default HTML format.

Note: Examples are not shown for SWT cache group metrics, local cache group metrics, dynamic global cache group metrics, grid metrics, or latch metrics.

To include latch metrics, you must enable them for the database, using the `ttXactAdmin` utility as follows:

```
% ttXactAdmin -latchstats on DSN
```

Summary Figure 3–1 shows most of a report summary. The summary is good for a quick look at database metrics, with further details provided in the subsequent tables. It includes the following sections:

- Memory Usage and Connections: This information includes information about memory usage (the `db.size` metrics) and connections established (the

connections.established metrics), including the number of client/server connections and direct connections. Any nonzero value for connections.established.threshold_exceeded, indicates too many connections.

- Load Profile: This gives an idea of the workload, showing the number of checkpoints, sorts (such as for ORDER BY statements), log buffer waits (delays when the log buffer fills and flushes to disk), inserts, updates, deletes, parses (such as for prepares), commits, and rollbacks. Consider whether there may be too many parses or too many durable commits (which are more expensive than non-durable commits).
- Instance Efficiency Percentage: Command Cache Hit %, Non-Parse/Execs %, Lock Hit %, and Log Buffer No Wait % are shown. All should be near 100%.
  - Lock Hit % estimates the percentage of lock requests that are granted without waiting.
  - Non-Parse/Execs % represents the percentage of SQL statement executions that do not require a prepare or reprepare.
  - Command Cache Hit % estimates the percentage of executions of SQL commands that can be found in the command cache.
  - Log Buffer No Wait % estimates the percentage of log insertions that do not have to wait due to log buffer waits.

Figure 3–1 ttStats report: summary

Summary

Info	Snap Id	Snap Time
Begin Snap:	3	2013-03-18 15:02:11.000000
End Snap:	4	2013-03-18 15:05:52.000000
Elapsed Time:		221 secs

Memory Usage and Connections

Metrics	Begin Value	End Value
connections.established.client_server	0	0
connections.established.count	20	20
connections.established.direct	20	20
connections.established.first.count	2	2
connections.established.threshold_exceeded	0	0
db.size.perm_allocated.kb	1048576	1048576
db.size.perm_high_water_mark.kb	183548	183746
db.size.perm_in_use.kb	183513	183711
db.size.temp_allocated.kb	524288	524288
db.size.temp_high_water_mark.kb	113468	115330
db.size.temp_in_use.kb	112616	112933

Load Profile

Metrics	Per Second	Per Transaction
Ckpt Size	1819680.4	62
Sorts	.2	0
Log Bytes	13030771.9	445
Log Buf Waits	0	0
Log Reads for Commit	0	0
Log Reads	0	0
Log Writes	41.9	0
Deletes	0	0
Inserts	6.3	0
Select	.9	0
Updates	0	0
Hard Parses	.2	0
Total Parses	.5	0
Durable Commit	.1	0
Non-durable Commit	29309.9	1
Rollback	0	0

Rows Per Read	0	Rows Per Write	0
Temp Indexes Created	0	Fast Path Log Buffer %	0

Instance Efficiency Percentage (Target 100%)

Command Cache Hit %	97.75	Non-Parse/Execs %	93.07
Lock Hit %	100	Log Buffer No Wait %	100

Statement statistics Figure 3–2 shows statement metrics from a report. Both external metrics (`stmt.executes`, `stmt.prepares`, and `stmt.reprepares` metrics) and internal metrics (`zzinternal` metrics) are shown. External metrics are generally of more interest. The `stmt.executes.count` value is the sum of all the other `stmt.executes` values.

Figure 3–2 ttStats report: statement statistics

Statement Statistics

Statistics	Value	Rate (Per Second)	Source
stmt.executes.alters	0	0	SystemStats table
stmt.executes.count	4738	592.25	SystemStats table
stmt.executes.creates	0	0	SystemStats table
stmt.executes.deletes	1	.13	SystemStats table
stmt.executes.drops	0	0	SystemStats table
stmt.executes.inserts	2475	309.38	SystemStats table
stmt.executes.merges	0	0	SystemStats table
stmt.executes.selects	2240	280	SystemStats table
stmt.executes.updates	16	2	SystemStats table
stmt.prepares.command_cache_miss	5	.63	SystemStats table
stmt.prepares.count	20	2.5	SystemStats table
stmt.reprepares.automatic	0	0	SystemStats table
stmt.reprepares.count	0	0	SystemStats table

Transaction statistics Figure 3–3 shows transaction metrics from a report. The `txn.commits.count` value is the sum of the `txn.commits.durable` and `txn.commits.nondurable` values. Other metrics shown are subsets of these metrics.

Figure 3–3 ttStats report: transaction statistics

Transaction Statistics

Statistics	Value	Rate (Per Second)	Source
txn.commits.count	128	16	SystemStats table
txn.commits.durable	1	.13	SystemStats table
txn.commits.internal.replication	120	15	SystemStats table
txn.commits.internal.xla	0	0	SystemStats table
txn.commits.nondurable	127	15.88	SystemStats table
txn.commits.replicated.durable	0	0	SystemStats table
txn.commits.replicated.nondurable	0	0	SystemStats table
txn.rollbacks	0	0	SystemStats table
zzinternal.parawt.txn.count	0	0	SystemStats table
zzinternal.parawt.txn.with.dependencies	0	0	SystemStats table
zzinternal.repl.par.txn.count	0	0	SystemStats table

SQL statistics: sort by executions Figure 3–4 shows an excerpt of SQL execution metrics from the SQL Statistics section of a report. When you look at the "sort by executions" metrics and "sort by preparations" metrics (shown in the next section), note which statements are used a lot and the number of preparations and the number of executions for each statement. Ideally, a statement is not prepared many times.

Figure 3–4 ttStats report: SQL execution statistics

SQL Statistics

SQL Sort by Executions
- Only top 30 SQL Commands are displayed

Executions	% Total	Cmd ID	Cmd Text	Source
2190	43.61	230938584	SELECT DESC_ID FROM SNAPSHOT_DESCRIPTION WHERE TRI	ttSQLCmdCacheInfo
2176	43.33	230333008	INSERT INTO SNAPSHOT_VALUE_PARAWT VALUES(:B4 , :B	ttSQLCmdCacheInfo
142	2.83	230873952	INSERT INTO SNAPSHOT_DESCRIPTION VALUES(:B2 , :B1	ttSQLCmdCacheInfo
142	2.83	230939608	SELECT MAX(DESC_ID) FROM SNAPSHOT_DESCRIPTION	ttSQLCmdCacheInfo
136	2.71	231387616	INSERT INTO SNAPSHOT_VALUE_SQL VALUES(:B6 , :B1 ,	ttSQLCmdCacheInfo
65	1.29	230071664	INSERT INTO SNAPSHOT_VALUE_CONFIG VALUES(:B3 , :B	ttSQLCmdCacheInfo
14	.28	230593168	INSERT INTO SNAPSHOT_VALUE_LOGHOLD VALUES(:B5 , :B	ttSQLCmdCacheInfo
12	.24	156087536	select name, owner, minval, maxval, increment, isr	ttSQLCmdCacheInfo
12	.24	227961232	update ttrep.reppeers set sendlsnhigh = :h, sendls	ttSQLCmdCacheInfo
12	.24	230203424	SELECT MAX(REPLPEER_ID) FROM SNAPSHOT_REPL_PEER	ttSQLCmdCacheInfo
12	.24	230207168	INSERT INTO SNAPSHOT_REPL_PEER VALUES(:B2 , :B1)	ttSQLCmdCacheInfo
12	.24	230466592	SELECT REPLPEER_ID FROM SNAPSHOT_REPL_PEER WHERE R	ttSQLCmdCacheInfo
12	.24	231459520	INSERT INTO SNAPSHOT_VALUE_REPL VALUES(:B8 , :B1	ttSQLCmdCacheInfo
10	.2	231482808	INSERT INTO SNAPSHOT_VALUE_PLSQL VALUES(:B3 , :B1	ttSQLCmdCacheInfo

SQL statistics: sort by preparations Figure 3–5 shows an excerpt of SQL preparation metrics from the SQL Statistics section of a report. Refer to the discussion in the preceding "sort by executions" section.

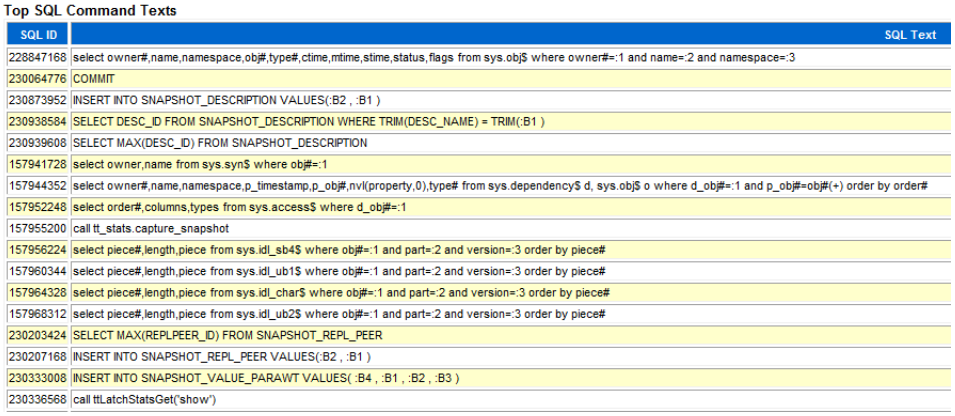
Figure 3–5 ttStats report: SQL preparation statistics

SQL Sort by Preparations
- Only top 30 SQL Commands are displayed

Preparations	% Total	Cmd ID	Cmd Text	Source
9	19.15	228847168	select owner#,name,namespace,obj#,type#,ctime,mtim	ttSQLCmdCacheInfo
2	4.26	230064776	COMMIT	ttSQLCmdCacheInfo
2	4.26	230873952	INSERT INTO SNAPSHOT_DESCRIPTION VALUES(:B2 , :B1	ttSQLCmdCacheInfo
2	4.26	230938584	SELECT DESC_ID FROM SNAPSHOT_DESCRIPTION WHERE TRI	ttSQLCmdCacheInfo
2	4.26	230939608	SELECT MAX(DESC_ID) FROM SNAPSHOT_DESCRIPTION	ttSQLCmdCacheInfo
1	2.13	157941728	select owner,name from sys.syn$ where obj#=:1	ttSQLCmdCacheInfo
1	2.13	157944352	select owner#,name,namespace,p_timestamp,p_obj#,nv	ttSQLCmdCacheInfo
1	2.13	157952248	select order#,columns,types from sys.access$ where	ttSQLCmdCacheInfo
1	2.13	157955200	call tt_stats.capture_snapshot	ttSQLCmdCacheInfo
1	2.13	157956224	select piece#,length,piece from sys.idl_sb4$ where	ttSQLCmdCacheInfo
1	2.13	157960344	select piece#,length,piece from sys.idl_ub1$ where	ttSQLCmdCacheInfo
1	2.13	157964328	select piece#,length,piece from sys.idl_char$ wher	ttSQLCmdCacheInfo
1	2.13	157968312	select piece#,length,piece from sys.idl_ub2$ where	ttSQLCmdCacheInfo
1	2.13	230203424	SELECT MAX(REPLPEER_ID) FROM SNAPSHOT_REPL_PEER	ttSQLCmdCacheInfo

SQL statistics: command texts Figure 3–6 shows an excerpt of SQL statements from the SQL Statistics section of a report. This report shows the complete text of each statement listed in the preceding "sort by executions" and "sort by preparations" reports, where longer statements are abbreviated.

Figure 3–6 ttStats report: SQL command texts

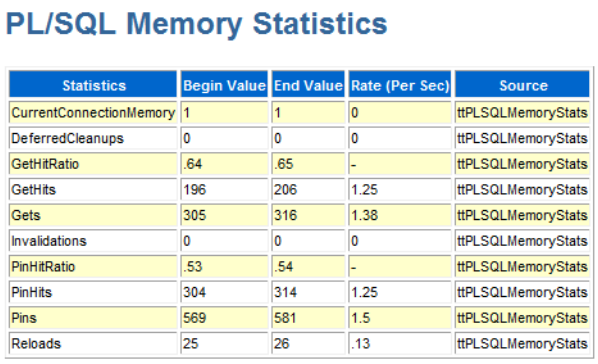


Top SQL Command Texts

SQL ID	SQL Text
228847168	select owner#,name,namespace,obj#,type#,ctime,mtime,stime,status,flags from sys.obj$ where owner#=:1 and name=:2 and namespace=:3
230064776	COMMIT
230873952	INSERT INTO SNAPSHOT_DESCRIPTION VALUES(:B2 , :B1)
230938584	SELECT DESC_ID FROM SNAPSHOT_DESCRIPTION WHERE TRIM(DESC_NAME) = TRIM(:B1)
230939608	SELECT MAX(DESC_ID) FROM SNAPSHOT_DESCRIPTION
157941728	select owner,name from sys.syn$ where obj#=:1
157944352	select owner#,name,namespace,p_timestamp,p_obj#,nvl(property,0),type# from sys.dependency$ d, sys.obj$ o where d_obj#=:1 and p_obj#=obj#(+) order by order#
157952248	select order#,columns,types from sys.access$ where d_obj#=:1
157955200	call tt_stats.capture_snapshot
157956224	select piece#,length,piece from sys.idl_sb4$ where obj#=:1 and part=:2 and version=:3 order by piece#
157960344	select piece#,length,piece from sys.idl_ub1$ where obj#=:1 and part=:2 and version=:3 order by piece#
157964328	select piece#,length,piece from sys.idl_char$ where obj#=:1 and part=:2 and version=:3 order by piece#
157968312	select piece#,length,piece from sys.idl_ub2$ where obj#=:1 and part=:2 and version=:3 order by piece#
230203424	SELECT MAX(REPLPEER_ID) FROM SNAPSHOT_REPL_PEER
230207168	INSERT INTO SNAPSHOT_REPL_PEER VALUES(:B2 , :B1)
230333008	INSERT INTO SNAPSHOT_VALUE_PARAWT VALUES(:B4 , :B1 , :B2 , :B3)
230336568	call ttLatchStatsGet('show')

PL/SQL memory statistics Figure 3–7 shows PL/SQL memory metrics from a report. These are metrics from the `ttPLSQLMemoryStats` built-in procedure. There should not be a significant difference between the start and end values of `GetHitRatio` or `PinHitRatio`.

Figure 3–7 ttStats report: PL/SQL memory statistics



PL/SQL Memory Statistics

Statistics	Begin Value	End Value	Rate (Per Sec)	Source
CurrentConnectionMemory	1	1	0	ttPLSQLMemoryStats
DeferredCleanups	0	0	0	ttPLSQLMemoryStats
GetHitRatio	.64	.65	-	ttPLSQLMemoryStats
GetHits	196	206	1.25	ttPLSQLMemoryStats
Gets	305	316	1.38	ttPLSQLMemoryStats
Invalidations	0	0	0	ttPLSQLMemoryStats
PinHitRatio	.53	.54	-	ttPLSQLMemoryStats
PinHits	304	314	1.25	ttPLSQLMemoryStats
Pins	569	581	1.5	ttPLSQLMemoryStats
Reloads	25	26	.13	ttPLSQLMemoryStats

Replication statistics Figure 3–8 shows replication metrics from a report. For each transmitter (where there could be multiple transmitters per master), the metrics indicate advancement through the log, including how many records were sent to the receiver. `Repl_Peer` indicates the subscriber. `Repl_Log_Behind` and `Repl_Latency` are significant in indicating whether replication is keeping up with the database workload.

Figure 3–8 ttStats report: replication statistics

Replication Statistics

Repl_Log_Send_LSN		Repl_Log_Behind		Repl_RPS		Repl_TPS		Repl_Latency		Repl_Peer
Begin Value	End Value	Begin Value	End Value	Value	Rate (Per Sec)	Value	Rate (Per Sec)	Begin Value	End Value	
0/17687464	0/18578344	0	0	-1	-1	-1	-1	-1	-1	REPLDB2:0
0/17687464	0/17687464	0	0	-1	-1	-1	-1	-1	-1	REPLDB2:1
0/17687464	0/17687464	0	0	-1	-1	-1	-1	-1	-1	REPLDB2:2
0/17687464	0/17687464	0	0	-1	-1	-1	-1	-1	-1	REPLDB2:3
0/17687464	0/18578344	0	0	-1	-1	-1	-1	-1	-1	SUB1:0
0/17654696	0/18578344	0	0	-1	-1	-1	-1	-1	-1	SUB1:1
0/17654696	0/18578344	0	0	-1	-1	-1	-1	-1	-1	SUB1:2
0/17654696	0/18578344	0	0	-1	-1	-1	-1	-1	-1	SUB1:3
0/17687464	0/18578344	0	0	-1	-1	-1	-1	-1	-1	SUB2:0
0/17671080	0/18578344	0	0	-1	-1	-1	-1	-1	-1	SUB2:1
0/17671080	0/18578344	0	0	-1	-1	-1	-1	-1	-1	SUB2:2
0/17671080	0/18578344	0	0	-1	-1	-1	-1	-1	-1	SUB2:3

Parallel replication/AWT statistics Figure 3–9 shows an excerpt of parallel replication/AWT metrics from a report. `Repl_Peer` indicates the subscriber. When parallel replication/AWT is configured, if replication metrics (discussed in the previous section) indicate difficulty keeping up with the workload, parallel replication/AWT metrics may indicate why. Each value is an aggregate across all tracks, but you can click **Show Details** (at the end of the metrics table, not shown here) to see the data for each track. High values for track switching—"switchin" and "switchout" metrics—may indicate contention. High values for "waits" metrics are also problematic, indicating situations such as one transaction having to wait for a previous transaction to commit before it can begin or before it can commit.

Figure 3–9 ttStats report: parallel replication/AWT statistics

Parallel Replication/AWT Statistics

Statistics	MIN Value	MIN Rate	MAX Value	MAX Rate	AVG Value	AVG Rate	SUM Value	SUM Rate	Repl_Peer	Source
zzinternal.log.trackswitchin.busy	0	0	0	0	0	0	0	0	REPLDB1	v$repstats view
zzinternal.log.trackswitchin.busy	0	0	0	0	0	0	0	0	REPLDB2	v$repstats view
zzinternal.log.trackswitchin.busy	0	0	0	0	0	0	0	0	SUB1	v$repstats view
zzinternal.log.trackswitchin.busy	0	0	0	0	0	0	0	0	SUB2	v$repstats view
zzinternal.log.trackswitchin.timeout	0	0	1	.13	.25	.03	1	.13	REPLDB1	v$repstats view
zzinternal.log.trackswitchin.timeout	0	0	0	0	0	0	0	0	REPLDB2	v$repstats view
zzinternal.log.trackswitchin.timeout	0	0	0	0	0	0	0	0	SUB1	v$repstats view
zzinternal.log.trackswitchin.timeout	0	0	0	0	0	0	0	0	SUB2	v$repstats view
zzinternal.log.trackswitchout.busy	0	0	0	0	0	0	0	0	REPLDB1	v$repstats view
zzinternal.log.trackswitchout.busy	0	0	0	0	0	0	0	0	REPLDB2	v$repstats view
zzinternal.log.trackswitchout.busy	0	0	0	0	0	0	0	0	SUB1	v$repstats view
zzinternal.log.trackswitchout.busy	0	0	0	0	0	0	0	0	SUB2	v$repstats view
zzinternal.log.trackswitchout.timeout	0	0	1	.13	.25	.03	1	.13	REPLDB1	v$repstats view
zzinternal.log.trackswitchout.timeout	0	0	0	0	0	0	0	0	REPLDB2	v$repstats view
zzinternal.log.trackswitchout.timeout	0	0	0	0	0	0	0	0	SUB1	v$repstats view
zzinternal.log.trackswitchout.timeout	0	0	0	0	0	0	0	0	SUB2	v$repstats view

Log statistics Figure 3–10 shows log metrics from a report. The report output notes that numbers in `log.file.earliest` and `log.file.latest` represent values in the begin and end snapshots. The `log.buffer.waits` metric is of particular interest. Log buffer waits occur when application processes cannot insert transaction data to the log buffer and must stall to wait for log buffer space to be freed. The usual reason for this is that the log flusher thread has not cleared out data fast enough. This may indicate that log buffer space is insufficient, disk bandwidth is insufficient, writing to disk is taking too long, or the log flusher is CPU-bound. (Also see "Managing transaction log buffers and files" and "Increase LogBufMB if needed" in *Oracle TimesTen In-Memory Database Operations Guide*.)

Figure 3–10 ttStats report: log statistics

Log Statistics

- **Numbers in log.file.earliest and log.file.latest represent values in begin snapshot and end snapshot**

Statistics	Value	Rate (Per Second)	Source
log.buffer.bytes_inserted	780960	97620	SystemStats table
log.buffer.insertions	6474	809.25	SystemStats table
log.buffer.waits	0	0	SystemStats table
log.commit.bytes.read	460168	57521	SystemStats table
log.commit.file.reads	0	0	SystemStats table
log.file.earliest	0	0	SystemStats table
log.file.latest	0	0	SystemStats table
log.file.reads	0	0	SystemStats table
log.file.writes	17	2.13	SystemStats table
log.files.generated	0	0	SystemStats table
log.forces	11	1.38	SystemStats table
log.last_log_lfn	0	0	SystemStats table
log.log_bytes_per_transaction	87	10.88	SystemStats table
log.recovery.bytes.read	0	0	SystemStats table
zzinternal.log.buffer.bytes_inserted.fast_path	0	0	SystemStats table
zzinternal.log.buffer.insertions.fast_path	0	0	SystemStats table
zzinternal.log.strand_switches.insertion_latch_held	0	0	SystemStats table
zzinternal.log.strand_switches.strand_full	0	0	SystemStats table
zzinternal.repl.transmitter.log.wait_sleeps	0	0	SystemStats table
zzinternal.repl.transmitter.log.waits	0	0	SystemStats table

Log holds Figure 3–11 shows log hold information from a report. It shows bookmark positions for checkpoint log holds for each checkpoint file, and bookmark positions for replication log holds for each replication subscriber. This report may also show log hold information for backup, XLA, and long-running transactions. Where the begin and end values are the same, there have been no movements.

Ideally there will be evidence of a smooth progression through the log file. (The `ttStats` monitor information may be more useful in tracking this.)

Figure 3–11 ttStats report: log holds

Log Holds

Statistics	Begin Value	End Value	Desc	Source
loghold.Checkpoint_hold_lsn	0/3796992	0/3796992	repldb1.ds0	ttLogHolds, ttBookmark
loghold.Checkpoint_hold_lsn	0/16197632	0/16197632	repldb1.ds1	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17687464	0/18578344	ADC6140793:REPLDB2:0	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17687464	0/17687464	ADC6140793:REPLDB2:1	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17687464	0/17687464	ADC6140793:REPLDB2:2	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17687464	0/17687464	ADC6140793:REPLDB2:3	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17687464	0/18578344	ADC6140793:SUB1:0	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17654696	0/18578344	ADC6140793:SUB1:1	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17654696	0/18578344	ADC6140793:SUB1:2	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17654696	0/18578344	ADC6140793:SUB1:3	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17687464	0/18578344	ADC6140793:SUB2:0	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17671080	0/18578344	ADC6140793:SUB2:1	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17671080	0/18578344	ADC6140793:SUB2:2	ttLogHolds, ttBookmark
loghold.Replication_hold_lsn	0/17671080	0/18578344	ADC6140793:SUB2:3	ttLogHolds, ttBookmark

Checkpoint statistics Figure 3–12 shows checkpoint metrics from a report.

Figure 3–12 ttStats report: checkpoint statistics

CheckPoint Statistics

Statistics	Value	Rate (Per Second)	Source
ckpt.bytes_written(MB)	0	0	SystemStats table
ckpt.bytes_written.during_recovery(MB)	0	0	SystemStats table
ckpt.completed	0	0	SystemStats table
ckpt.completed.fuzzy	0	0	SystemStats table
ckpt.writes	0	0	SystemStats table

Cache group statistics: AWT cache groups Figure 3–13 shows AWT cache group metrics from a report. Values are aggregates across all AWT cache groups. Information includes the number of calls to the Oracle database; the number of commits, rollbacks, and retries on Oracle; and the number of rows inserted, deleted, and updated by PL/SQL operations and by SQL operations.

Figure 3–13 ttStats report: AWT cache group statistics

Cache Group Statistics

AWT Cache Group Statistics

Statistics	Value	Rate(Per Sec)	Rate(Per AWT Txn)	Source
cg.awt.calls_to_oracle	330	15.71	.01	sys.SystemStats table
cg.awt.commits_on_oracle	330	15.71	.01	sys.SystemStats table
cg.awt.plsql_mode.batches	330	15.71	.01	sys.SystemStats table
cg.awt.plsql_mode.bytes	4587698	218461.81	98.6	sys.SystemStats table
cg.awt.plsql_mode.deletes.rows	0	0	0	sys.SystemStats table
cg.awt.plsql_mode.inserts.rows	0	0	0	sys.SystemStats table
cg.awt.plsql_mode.updates.rows	46530	2215.71	1	sys.SystemStats table
cg.awt.retries_on_oracle	0	0	0	sys.SystemStats table
cg.awt.rollbacks_on_oracle	0	0	0	sys.SystemStats table
cg.awt.sql_mode.batches	0	0	0	sys.SystemStats table
cg.awt.sql_mode.bytes	0	0	0	sys.SystemStats table
cg.awt.sql_mode.deletes.batches	0	0	0	sys.SystemStats table
cg.awt.sql_mode.deletes.rows	0	0	0	sys.SystemStats table
cg.awt.sql_mode.inserts.batches	0	0	0	sys.SystemStats table
cg.awt.sql_mode.inserts.rows	0	0	0	sys.SystemStats table
cg.awt.sql_mode.updates.batches	0	0	0	sys.SystemStats table
cg.awt.sql_mode.updates.rows	0	0	0	sys.SystemStats table
cg.awt.tt_txns	46530	2215.71	1	sys.SystemStats table
zzinternal.cg.awt.plsql_mode.exec_time	0	0	0	sys.SystemStats table
zzinternal.cg.awt.rxbatches	0	0	0	sys.SystemStats table
zzinternal.cg.awt.rxskips	0	0	0	sys.SystemStats table
zzinternal.cg.awt.sql_mode.exec_time	0	0	0	sys.SystemStats table
zzinternal.cg.awt.tt_proc_time	0	0	0	sys.SystemStats table

Cache group statistics: auto-refresh cache groups Figure 3–14 shows auto-refresh cache group metrics from a report. Values are aggregates across all auto-refresh cache groups. Whether cache groups are in full or incremental refresh mode is reflected by the `cg.autorefresh.full_refreshes` value with respect to the `cg.autorefresh.cycles.completed` value (which indicates the total number of refreshes).

Figure 3–14 ttStats report: auto-refresh cache group statistics

Auto-refresh Cache Group Stats

Statistics	Value	Rate(Per Sec)	Rate(Per Cycle)	Source
cg.autorefresh.cycles.completed	5	.24	1	sys.SystemStats table
cg.autorefresh.cycles.failed	0	0	0	sys.SystemStats table
cg.autorefresh.deletes.rows	0	0	0	sys.SystemStats table
cg.autorefresh.full_refreshes	5	.24	1	sys.SystemStats table
cg.autorefresh.inserts.rows	0	0	0	sys.SystemStats table
cg.autorefresh.updates.rows	0	0	0	sys.SystemStats table
zzinternal.cg.autorefresh.bookmark_cycles	0	0	0	sys.SystemStats table
zzinternal.cg.autorefresh.bookmark_updates	0	0	0	sys.SystemStats table
zzinternal.cg.autorefresh.garbage_collector_cycles	0	0	0	sys.SystemStats table
zzinternal.cg.autorefresh.log_table.rows.garbage_collected	0	0	0	sys.SystemStats table
zzinternal.cg.autorefresh.log_table.rows.marked	0	0	0	sys.SystemStats table
zzinternal.cg.autorefresh.marker_cycles	0	0	0	sys.SystemStats table

Database activity statistics Figure 3–15 shows an excerpt of database activity metrics from a report—index activity, memory activity, and table activity. For hash indexes and range indexes, information includes deletes, inserts, rows fetched, and scans. For memory usage, it shows size data. For tables, it shows rows read, deleted, inserted, and updated.

Figure 3–15 ttStats report: database activity statistics

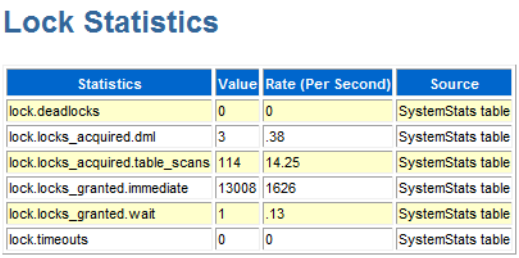
DB Activity Statistics

Statistics	Value	Rate (Per Second)	Source
db.cache_hits	0	0	SystemStats table
db.index.hash.deletes	0	0	SystemStats table
db.index.hash.inserts	0	0	SystemStats table
db.index.hash.inserts.recovery_rebuild	0	0	SystemStats table
db.index.hash.rows_fetched.count	0	0	SystemStats table
db.index.hash.rows_fetched.repl	0	0	SystemStats table
db.index.hash.scans.count	0	0	SystemStats table
db.index.hash.scans.repl	0	0	SystemStats table
db.index.range.deletes	0	0	SystemStats table
db.index.range.inserts.count	0	0	SystemStats table
db.index.range.inserts.recovery_rebuild	0	0	SystemStats table
db.index.range.rows_fetched.count	0	0	SystemStats table
db.index.range.rows_fetched.repl	0	0	SystemStats table
db.index.range.scans.count	0	0	SystemStats table
db.index.range.scans.repl	0	0	SystemStats table
db.index.range.updates	0	0	SystemStats table
db.index.rebuilds	0	0	SystemStats table
db.index.temporary.created	0	0	SystemStats table
db.index.temporary.rows_fetched.count	0	0	SystemStats table
db.index.temporary.rows_fetched.repl	0	0	SystemStats table
db.index.temporary.scans.count	0	0	SystemStats table
db.index.temporary.scans.repl	0	0	SystemStats table
db.joins.merge	0	0	SystemStats table
db.joins.nested_loop	28	3.5	SystemStats table
db.passthroughs	0	0	SystemStats table
db.size.perm_allocated.kb	0	0	SystemStats table
db.size.perm_high_water_mark.kb	595	74.38	SystemStats table
db.size.perm_in_use.kb	595	74.38	SystemStats table
db.size.temp_allocated.kb	0	0	SystemStats table
db.size.temp_high_water_mark.kb	30	3.75	SystemStats table
db.size.temp_in_use.kb	886	110.75	SystemStats table
db.sorts	4	.5	SystemStats table
db.table.full_scans	17	2.13	SystemStats table

Lock statistics Figure 3–16 shows lock metrics from a report. This provides information about deadlocks, locks acquired, locks granted, and lock timeouts. In

particular, `lock.deadlocks`, `lock.locks_granted.wait`, and `lock.timeouts` may indicate lock contention.

Figure 3–16 ttStats report: lock statistics

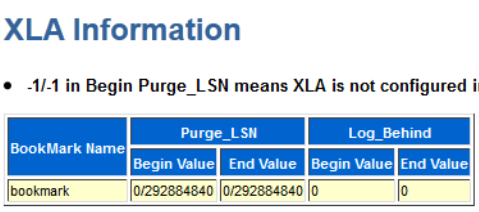


Lock Statistics

Statistics	Value	Rate (Per Second)	Source
lock.deadlocks	0	0	SystemStats table
lock.locks_acquired.dml	3	.38	SystemStats table
lock.locks_acquired.table_scans	114	14.25	SystemStats table
lock.locks_granted.immediate	13008	1626	SystemStats table
lock.locks_granted.wait	1	.13	SystemStats table
lock.timeouts	0	0	SystemStats table

XLA information Figure 3–17 shows XLA bookmark information from a report. For each bookmark, the begin and end values are shown for `Purge_LSN`, which indicates the position in the log file prior to which information has been purged, and for `Log_Behind`, which indicates whether there is a lag between the position of the XLA transaction and the position of the most recent log file.

Figure 3–17 ttStats report: XLA information



XLA Information

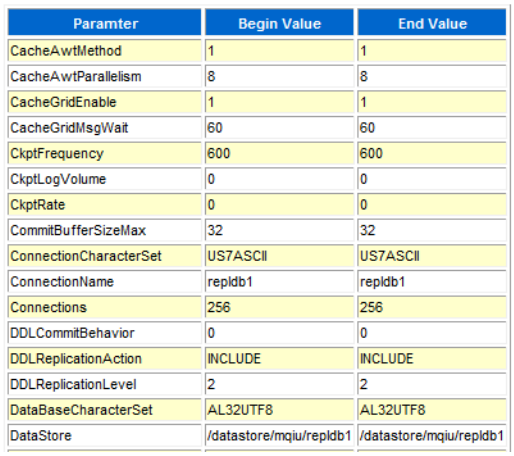
- -1/-1 in Begin Purge_LSN means XLA is not configured in begin_snapshot.

BookMark Name	Purge_LSN		Log_Behind	
	Begin Value	End Value	Begin Value	End Value
bookmark	0/292884840	0/292884840	0	0

Configuration parameters Figure 3–18 shows database configuration parameter settings from a report. For reference, each report shows the begin and end values of each TimesTen connection attribute.

For information about connection attributes, see Chapter 1, "Connection Attributes".

Figure 3–18 ttStats report: configuration parameters



Configuration Parameters

Paramter	Begin Value	End Value
CacheAwtMethod	1	1
CacheAwtParallelism	8	8
CacheGridEnable	1	1
CacheGridMsgWait	60	60
CkptFrequency	600	600
CkptLogVolume	0	0
CkptRate	0	0
CommitBufferSizeMax	32	32
ConnectionCharacterSet	US7ASCII	US7ASCII
ConnectionName	repldb1	repldb1
Connections	256	256
DDLCommitBehavior	0	0
DDLReplicationAction	INCLUDE	INCLUDE
DDLReplicationLevel	2	2
DataBaseCharacterSet	AL32UTF8	AL32UTF8
DataStore	/datastore/mqiu/repldb1	/datastore/mqiu/repldb1

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 connections 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 *Oracle TimesTen In-Memory Database Operations Guide*.
- Miscellaneous status information.

If you specify a connection string or DSN, `ttStatus` outputs only the information for the specified database.

Required privilege

This utility requires no privileges.

Syntax

```
ttStatus {-h | -help | -?}
ttStatus {-V | -version}
ttStatus [-v] [-r secs] [-[no]pretty]
ttStatus [-r secs] [-[no]pretty] {DSN | -connStr connection_string | dspath}
```

Options

`ttStatus` has the options:

Option	Description

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-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 for which to get a status.
`-dsn` *`DSN`*	Specifies the DSN for which you want status. If no DSN is specified, shows the status of all connections to the data store.

Option	Description
-[no]pretty	With [no], indicates that pretty formatting is not used. The default is pretty formatting, which uses the values of the ConnectionName connection attribute.
-r *secs*	Enables 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 call the procedure, a report that describes the current state of the system is displayed to stdout. To get the status for the cachedb1_1122 DSN, use the following:

```
ttstatus cachedb1_1122

TimesTen status report as of Thu May 02 19:45:43 2013

Daemon pid 5280 port 53392 instance tt1122_32
TimesTen server pid 3940 started on port 53393
------------------------------------------------------------------------
Data store cachedb1_1122
There are 12 connections to the data store
Shared Memory KEY Global\cachedb1_1122.c|. . .HANDLE 0x254
PL/SQL Memory KEY Global\cachedb1_1122.c|. . . HANDLE 0x258 Address 0x5B8C0000
Type            PID     Context     Connection Name              ConnID
Process         5196    0x01066a58  cachedb1_1122                     1
Subdaemon       3912    0x00b2c398  Manager                        2047
Subdaemon       3912    0x00b7e4a0  Rollback                       2046
Subdaemon       3912    0x015d25e8  Flusher                        2045
Subdaemon       3912    0x015e46b0  Monitor                        2044
Subdaemon       3912    0x016767f8  Deadlock Detector              2043
Subdaemon       3912    0x016888c0  Checkpoint                     2041
Subdaemon       3912    0x0d350578  Aging                          2042
Subdaemon       3912    0x0d362640  Log Marker                     2040
Subdaemon       3912    0x0d4347c8  AsyncMV                        2039
Subdaemon       3912    0x0d446890  HistGC                         2038
Subdaemon       3912    0x0d458958  IndexGC                        2037
Replication policy  : Manual
Cache Agent policy  : Manual
PL/SQL enabled.
------------------------------------------------------------------------
Accessible by group . . .
End of report
```

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.

The ttStatus utility only reports the RAM policy if it is not inUse.

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 TimesTen writes all messages 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.

Notes

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 3-144 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.

Examples

```
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 *Oracle TimesTen In-Memory Database Troubleshooting 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 *Oracle TimesTen In-Memory Database Troubleshooting 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 connection 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 connection 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 administrator, instance home directory, daemon home directory, port number and build timestamp. You can specify various levels of output:

- You can specify `ttVersion` with no options to list abbreviated output.
- You can specify the `-m` option to list enhanced output.
- You can specify an attribute to list output only for a specific attribute.

Required privilege

This utility requires no privileges.

Syntax

```
ttVersion [-m] [attribute] [...]
```

Options

`ttVersion` has the option:

Option	Description
`-m`	Generates computer-readable enhanced output. If not specified and no attribute is specified, abbreviated information is output.
attribute	Generates information only about the specified attribute. You can specify multiple attributes. When you specify more than one attribute, the output is displayed with an equal sign after the attribute name.

Attributes

`ttVersion` has these attributes:

Attribute	Description
`patched`	Lists `yes` or `no`, indicating whether the release has been patched.
`product`	Lists the name of the product.
`major1`	The first number of the five-place release number.
`major2`	The second number of the five-place release number.
`major3`	The third number of the five-place release number.
`patch`	The fourth number of the five-place release number.
`portpatch`	The fifth number of the five-place release number.
`version`	All five numbers of the release number, separated by periods.
`shortversion`	The first three numbers of the five-place release number.
`numversion`	The first four numbers of the five-place release number, represented by three digits for each place.

Attribute	Description
bits	Lists 32 or 64 to indicate the bit-level of the operating system for which this release is intended.
os	The operating system for which this release is intended
buildstamp	A number indicating the specific build.
buildtime	The UTC time the release was built, for example: 2013-03-19T17:21:59Z
clientonly	Lists yes or no to indicate if the release is a client-only release
instance	The name of the instance, for example: tt1122_32.
effective_port	The number of the port on which the main daemon listens.
orig_port	The original number of the port on which the main daemon listened.
instance_admin	The user name of the instance administrator.
effective_insthome	The path that indicates the location of the instance.
effective_insthome_long	On Windows, the path that indicates the location of the instance including a bit extension on the instance name.
orig_insthome	The path that indicates the location of the instance.
effective_daemonhome	The path to the home of the daemon for the specific instance.
effective_daemonhome_long	On Windows, the path to the home of the daemon for the specific instance, including a bit extension on the instance name.
orig_daemonhome	The path to the original home of the daemon.
plsql	Indicates if PL/SQL is configured for this instance. 0 indicates that PL/SQL is not configured. 1 indicates that PL/SQL is configured. The value corresponds with the setting of the PLSQL connection attribute.
group_name	The name of the instance group.

Output

ttVersion produces the following sample output.

```
TimesTen Release 11.2.2 (32 bit Linux/x86) (tt1122_32:53384)
2013-03-26T23:00:04Z
Instance admin: terry
Instance home directory: spider/terry/TimesTen/tt1122_32
Daemon home directory: spider/terry/TimesTen/tt1122_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=2
patch=5
portpatch=0
version=11.2.2.5.0
shortversion=1122
numversion=110200020500
bits=32
```

```
os=Linux
buildtstamp=1364278134
buildtime=2013-03-26T06:08:54Z
clientonly=no
instance=tt1122_32
effective_port=53392
orig_port=53392
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 enables you to heuristically commit, terminate or forget an XA transaction branch.

Applications should monitor log holds and the accumulation of log files. For more information, see "Monitoring accumulation of transaction log files" in the *Oracle TimesTen In-Memory Database Operations Guide*.

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 which privilege is needed, if any.

Syntax

```
ttXactAdmin {-h | -help | -?}
ttXactAdmin {-V | -version}

ttXactAdmin [-v verbosity] [-mt maxTrans] [-ml maxLocks] [-pid pid]
[-xact xid] [-tbl [owner.]tableName] [-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.

Option	Description
DSN	Indicates the ODBC data source name of the database to be administered. This option requires ADMIN privileges.
-h -help -?	Prints a usage message and exits.
-HAbort *xid*	Heuristically terminates 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.
-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 (default) - Shows the latch information, including access counts and other stats.
-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.

Option	Description
-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 (default) - Displays the names of the tables for row locks.
-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*	Enables you to roll back a transaction. This may be particularly useful for long running transactions. The 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 TimesTen Cache operations on the Oracle database. Operations include passthrough statements, flushing, manual loading, manual refreshing, synchronous writethrough, propagating, and dynamic 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 identify the connection of the program executing the transaction. The second part is a potentially large value (an unsigned integer), that distinguishes 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 terminating. 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 terminated 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 isolation. `IX` - 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 isolation. Lock "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.

Examples

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 11.2.2.0.0

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)*.

For databases, TimesTen only holds `S` locks when the isolation mode is serializable. For commands, `S` only means "shared" lock, and can be held in either serializable or read-committed isolation modes.

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}

ttXactLog [-v verbosity] -logAnalyze
[-s subscriberName -host hostname]]
[-xid xid] {-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/2011.ds0, /home/payroll/2011.ds1` and several transaction log files `/home/payroll/2011.log`*n*,dspath is `/home/payroll/2011`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

Option	Description
-headers *records*	Prints one header for every *records* records. A value of 0 disables headers.
-host *hostName*	Specifies the name of the host on which the subscriber resides. Use this option with the -subscriber option, if the name of the subscriber is ambiguous.
-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.
-l1	Considers this log record only (unless an -l2 argument is present).
-l2	Considers records between -l1 and -l2, inclusive.
-logAnalyze	Determines the remaining amount of a database to be replicated for one or all of the subscribers. Use with the -v option to print: 1 - A summary for every track (default). 2 - Level 1 plus a track analysis. 3 - Level 2 plus an in-depth transaction analysis. Use with -subscriber and -host to get information for a specific subscriber. Use with -xid to look for a specific transaction ID.
-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 4000).
-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.
-subscriber *subscriberName*	Specifies the name of the subscriber. To qualify the name of the subscriber, use -host *hostname*.
-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 (default) - Print log record headers too. 2 - Print log record bodies too, except long data. 3 - Print full log records (see -m option).
-x *xid*	Specifies the transaction ID.

Examples

```
ttXactLog -v 3 -m 100 /users/pat/TimesTen/Daily/F112697SS
```

Notes

This utility is supported only where the TimesTen Data Manager is installed.

See also

"Analyze outstanding transactions in the replication log" in the *Oracle TimesTen In-Memory Database Replication Guide*.

4

System Limits

The following sections list all TimesTen system limits and defaults.

System limits and defaults

Specific operating system limits may take precedence over these values. For more information, see "Installation prerequisites" in *Oracle TimesTen In-Memory Database Installation Guide*.

Description	32-bit Value	64-bit Value
Maximum number of subscriber databases in a replication scheme that is not an active standby pair.	128	128
Maximum number of propagators in a replication scheme. Each propagator can have the maximum number of subscribers.	128	128
Maximum number of subscriber databases in an active standby pair.	127	127
Minimum database size (bytes). Size includes both the permanent and temporary space required to perform operations on the database.	32 MB	32 MB
Maximum length for a fixed-length column (bytes).	8,300	8,300
Maximum number of columns in a table.	1,000	1,000
Maximum number of columns in an `ORDER BY` clause.	1,000	1,000
Maximum number of columns in a `GROUP BY` clause.	1,000	1,000

Description	32-bit Value	64-bit Value
Maximum cumulative length of a row's fixed-length columns (bytes).	32,768	32,768
Maximum length for a varying-length column (bytes).	2^{22} = 4,194,304	2^{22} = 4,194,304
Maximum length for a replicated column.	4 MB 16 MB for columns with `BLOB` type	4 MB 16 MB for columns with `BLOB` type
Maximum number of concurrent connections to a database (including system connections).	2047	2047
Maximum number of concurrent application connections to a database (may be limited by semaphore configuration or `Connections` DSN attribute or both).	2000	2000
Maximum number of connections (system and application) across all databases in an instance.	2048	2048
Maximum number of concurrent client connections to a TimesTen instance. **Note:** Some instances may support a slightly smaller maximum number of connections depending on such things as whether the database is shared or replicated and operating system limits. Most configurations support no less than 2,000 connections.	2048	2048
Maximum length of database names.	32	32
Maximum length of the path name for a database in an asynchronous writethrough cache group	248	248
Maximum number of projected expressions in a `SELECT` statement.	32,767	32,767
Maximum length of string specifying a join order.	1,024	1,024
Maximum number of columns in an index (or primary key).	16	16
Maximum length of basic names.	30	30
Maximum length of displayed predicate string in the `SYS.PLAN` table.	1,024	1,024

Description	32-bit Value	64-bit Value
Maximum length of SQL statement, including the `NULL` terminator.	409,600	409,600
Maximum number of table references in an SQL query.	24	24
Maximum number of indexes on a table.	500	500
Maximum number of partitions in a table.	999	999
Maximum number of prepared PL/SQL statements per connection.	5000	5000
Maximum number of recently-used PL/SQL blocks that can be cached per session.	5000	5000
Maximum number of concurrent shared memory segment client/server connections per TimesTen instance.	512	512
Maximum size of IPC shared memory segment for client/server connections	1 gigabyte	1 gigabyte
Maximum number of allocated statement handles per shared memory segment client/server connection.	512	512
Maximum depth of nesting subqueries.	Equal to the maximum number of table references in a SQL query.	Equal to the maximum number of table references in a SQL query.
Maximum error message length for applications that specify an error message length (for example, through a call to `SQLError`).	512	512

Limits on number of open files

Each process connected to a TimesTen database keeps at least one operating-system file descriptor open from the time of the first connection until the process terminates. Additional file descriptors may be opened for each database connection:

- Connections to databases that have logging to disk enabled require an additional two file descriptors for the duration of the connection.
- An additional file descriptor is needed for the duration of database checkpoints issued by the process.
- Additional file descriptors may be opened during transaction commit or rollback operations.

For multithreaded applications that maintain many concurrent TimesTen database connections, the default number of open files permitted to each process by the operating system may be too low.

- On Solaris, the default limit is 256 open files and may be raised for a session with the `ulimit` command (`limit` for `csh` users). You can also set the per-process limit programmatically with `setrlimit`.
- On AIX, the limit is 2,048 open files, so you are not likely to encounter problems.
- On Linux, the default limit is 1,024 open files, so you are not likely to encounter problems.
- On Windows, the default limit is at least 2,000 open files, so you are not likely to encounter problems.

Most of the open file descriptors are used for reading and writing database recovery log files. If a process fails to open a log file, the database is marked as requiring recovery and all current connections to the database are terminated.

Path names

TimesTen does not support file path names that contain multibyte characters. Ensure the installation path, database path, transaction log path, and temporary file path do not contain any multibyte characters.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X