Oracle® TimesTen In-Memory Database

Reference

Release 18.1

E61204-10

November 2020

ORACLE®

Contents

3 Built-In Procedures

4   TimesTen Scaleout Utilities

5 Utilities

6 System Limits

Index

Preface

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

- Oracle TimesTen In-Memory Database in classic mode, or TimesTen Classic, refers to single-instance and replicated databases (as in previous releases).

- Oracle TimesTen In-Memory Database in grid mode, or TimesTen Scaleout, refers to a multiple-instance distributed database. TimesTen Scaleout is a grid of interconnected hosts running instances that work together to provide fast access, fault tolerance, and high availability for in-memory data.

- TimesTen alone refers to both classic and grid modes (such as in references to TimesTen utilities, releases, distributions, installations, actions taken by the database, and functionality within the database).

- TimesTen Cache is ideal for caching performance-critical subsets of an Oracle database into cache tables within a TimesTen database for improved response time in the application tier. Cache tables can be read-only or updatable. Applications read and update the cache tables using standard Structured Query Language (SQL) while data synchronization between the TimesTen database and the Oracle database is performed automatically. TimesTen Cache offers all of the functionality and performance of TimesTen Classic, plus the additional functionality for caching Oracle Database tables.

- TimesTen Replication features, available with TimesTen Classic or TimesTen Cache, enable high availability.

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

Audience

This 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 at
https://docs.oracle.com/database/timesten-18.1.

Oracle Database documentation is also available on the Oracle documentation website. This may be especially useful for Oracle Database features that TimesTen supports but does not attempt to fully document, such as OCI and Pro*C/C++.

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. The term Linux is used separately. Refer to "Platforms and compilers" in Oracle TimesTen In-Memory Database Release Notes (README.html) in your installation directory for specific platform versions supported by TimesTen.

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

This document uses the following text conventions:

Convention	Meaning
italic	Italic type indicates terms defined in text, book titles, or emphasis.
monospace	Monospace type indicates code, commands, URLs, function names, attribute names, directory names, file names, text that appears on the screen, or text that you enter.
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: `LIBS = -Ltimesten_home/install/lib -ltten` Replace *timesten_home* with the path to the TimesTen instance home directory.
[ ]	Square brackets indicate that an item in a command line is optional.
{ }	Curly braces indicated that you must choose one of the items separated by a vertical bar (	) in a command line.
		A vertical bar (or pipe) separates alternative arguments.
. . .	An ellipsis (. . .) after an argument indicates that you may use more than one argument on a single command line. An ellipsis in a code example indicates that what is shown is only a partial example.
% or $	The percent sign or dollar sign indicates the UNIX shell prompt, depending on the shell that is used.
#	The number (or pound) sign indicates the UNIX root prompt.

In addition, TimesTen documentation uses the following special conventions.

Convention	Meaning
installation_dir	The path that represents the directory where TimesTen is installed.
timesten_home	The path that represents the home directory of a TimesTen instance.
release or *rr*	The first two parts in a release number, with or without the dot. The first two parts of a release number represent a major TimesTen release. For example, 181 or 18.1 represents TimesTen Release 18.1.
DSN	TimesTen data source name (for the TimesTen database).

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

Documentation Accessibility

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

Access to Oracle Support

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

What's New

This section summarizes the new features of TimesTen In-Memory Database Release 18.1 that are documented in this guide and provides links to more information.

New features in release 18.1.4.1.0

- In TimesTen Classic, if you are using a read-only cache group and have created a second TimesTen instance, then the client can fail over to this TimesTen instance. The cached data can be reloaded directly from the Oracle database. See "TTC_Server or TTC_Server1" on page 2-129, "TTC_Server2, TTC_ServerN" on page 2-131, "TTC_Server_DSN" on page 2-133, "TTC_Server_DSN2, TTC_Server_DSNn" on page 2-134, "TCP_Port" on page 2-120, and "TCP_Port2, TCP_PortN" on page 2-121 for more details.

- If any request to create a channel between elements hangs due to software issues or network failures, then all channel create requests could be blocked. You can set a timeout for how long to wait for a channel create request to a remote element. See "ChannelCreateTimeout" on page 2-51 for more information.

- To increase the performance of database import operations, the `ttGridAdmin dbImport` command now enables you to use multiple threads to import database objects with the use of the `-numThreads` option.

- You can enable Transport Layer Security (TLS) to provide encrypted network communication for all replication agent communication and all client/server communication. "CipherSuites," "Encryption," "SSLClientAuthentication," "Wallet," "SSLRenegotiationPeriod," and "SSLRenegotiationSize."

- This release of TimesTen Scaleout has a new type of backup: staged backups. This type of backup eliminates the overhead of creating local copies of the checkpoint and log files and reduces the network traffic of creating a remote copy in the repository. Staged backups are ideal for when you want to make regular backups on a second site that is independent to your main site. See "BackupFailThreshold" on page 2-28 for more information.

- You can now set a time interval for calculating the fragmentation percentage of the change log tables on the Oracle database. Use the `ttCacheConfig` built-in procedure providing the `AutorefreshLogMonitorInterval` as the value parameter.

New features in release 18.1.2.1.0

The following features are documented for the new "forced disconnect" feature, which forces all user connections to the specified database to be disconnected. This is useful, for example, prior to maintenance operations. Closing connections is mandatory to ensure a smooth shutdown and no data loss.

- `ForceDisconnectEnabled` connection attribute (TimesTen Classic). See "ForceDisconnectEnabled" on page 2-37 for details.

- `ttAdmin -disconnect` option (TimesTen Classic). See "Force disconnect" on page 5-11 for details.

- `ttGridAdmin dbDisconnect` and `dbDisconnectStatus` commands (TimesTen Scaleout). See "Force all connections to disconnect (dbDisconnect)" on page 4-35 for details.

These additional new features are documented:

- `ttGridAdmin dbDistribute -resync` option attempts to resynchronize metadata in the user database with metadata in the active management instance in case the state of a `dbDistribute -apply` command is unknown. "Set or modify the distribution scheme of a database (dbDistribute)" on page 4-39 for details.

- `PLSQL_SESSION_CACHED_CURSORS` connection attribute specifies the number of session cursors to cache. A user may adjust the setting to free up space not currently needed in the cache. See "PLSQL_SESSION_CACHED_CURSORS" on page 2-97 for details.

- `ttDBConfig` built-in procedure supports a new parameter, `PLSQL_OPEN_CURSORS`, which specifies the maximum number of PL/SQL cursors that can be open in a session at one time. See "ttDBConfig" on page 3-111 for details.

- `ttCkptHistory` built-in procedure returns new columns: *reason*, *logsPurged*, *bookmarkName*, and *additional_details*. See "ttCkptHistory" on page 3-92 for details.

- `ttInstallDSN` utility, for TimesTen Scaleout, generates a Windows client DSN for each of one or more entries in the provided input file and installs them into the ODBC control panel as system DSNs. See "ttInstallDSN" on page 5-57 for details.

New features in release 18.1.1.1.0

This release of TimesTen contains a configuration file associated with each instance. See Chapter 1, "TimesTen Instance Configuration File" for details about this file.

The book contains descriptions of new and changed utilities:

- `ttGridAdmin` utility
- `ttGridRollout` utility
- `ttInstallationCheck` utility
- `ttInstallDSN` utility
- `ttInstanceCreate` utility
- `ttInstanceDelete` utility
- `ttInstanceModify` utility
- Changes to `ttXactAdmin` utility
- Changes to `ttBulkCp` utility
- Changes to `ttStats` utility

The book contains descriptions of new attributes:

- `TTC_TCP_KEEPALIVE_TIME_MS`, `TTC_TCP_KEEPALIVE_INTVL_MS`, `TTC_TCP_KEEPALIVE_PROBES`, `TTC_Random_Selection`, `TTC_REDIRECT`, `TTC_Redirect_Limit` and `TTC_RollbackRequiredOnFailover` connection attributes

- Durability connection attribute

- EpochInterval and CreateEpochAtCommit connection attributes

- IncludeInCore connection attribute

- OptimizerHint connection attribute

- Changes to ttStats utility

The book contains descriptions of new built-in procedures:

- ttCacheADGStandbyStateGet

- ttCacheADGStandbyStateSet

- ttCacheADGStandyTimeoutGet

- ttCacheADGStandbyTimeoutSet

- ttEpochCreate

- ttEpochSessionGet

- ttHeapinfo

- ttLatchStatsGet

xx

1

TimesTen Instance Configuration File

This chapter describes the TimesTen configuration file attributes. The configuration files contains the attributes of the TimesTen instance and their values.

The `timesten.conf` file is located at *timesten_home*/conf/timesten.conf.

Each line of the configuration file consists of *one name=value* pair.

In TimesTen Classic, you can change values through various TimesTen utilities or connection attributes, at instance creation time or through editing this file. For more information about changing values by editing this file, see "Working with the TimesTen Daemon" in the *Oracle TimesTen In-Memory Database Operations Guide*.

In TimesTen Scaleout, you can change the values in this file by using the `ttGridAdmin instanceConfigImport` command. For more details, see "Import instance configuration attributes (instanceConfigImport)" on page 4-82. Do not edit the configuration file by hand.

Configuration attributes

Some attributes in the configuration file are required. Other attributes are optional.

Required attributes

The configuration file contains these required attributes:

Attribute name	Description	Attribute value and type	Default Value
instance_name	The name of the TimesTen instance. This attribute is optional for client only instances.	An ASCII name from 1 to 255 characters long.	instance1
daemon_port	The port number on which the TimesTen daemon for this instance listens. TimesTen ignores this attribute for client only instances.	A valid TCP port number.	6624
admin_user	The OS user name of the instance administrator that matches the OS owner of the instance home directory. This entry is added by the ttInstanceCreate utility and must not be changed.	String	None

Attribute name	Description	Attribute value and type	Default Value
admin_uid	The OS uid number of the instance administrator. This entry is added by the ttInstanceCreate utility and must not be changed.	Integer	None
hostname	The OS hostname of the instance. This entry is added by the ttInstanceCreate utility, and must not be changed.	String	None
instance_guid	A globally unique ID, guaranteed to be different than the ID of any other instance in the world. TimesTen adds this ID during instance creation.	String	None

Optional attributes

The configuration file can contain these optional attributes:.

Attribute name	Description	Attribute value and type	Default
timesten_release	The release that created the instance home and that can use the instance home.	A TimesTen release number.	The installed TimesTen release number.
client_only	Indicates if the instance only supports client operations. If yes, the instance supports only client operations. This entry is added by the ttInstanceCreate utility and must not be changed.	Yes or No	No
tns_admin	In a TimesTen Cache environment, the directory that contains the TNS_ADMIN settings to allow TimesTen to communicate with the Oracle database. This attribute is required in a TimesTen Cache environment only. This entry is added by the ttInstanceCreate utility and can be changed using the ttInstanceModify utility.	A directory	None

Attribute name	Description	Attribute value and type	Default
listen_addr and listen_6_addr	listen_addr indicates the IPv4 address where the TimesTen daemon and servers listen. listen_6_addr indicates the IPv6 address where the TimesTen daemon and servers listen. By default, TimesTen supports IPv6.	String	None
enableipv6	Indicates if TimesTen is configured to support IPv6.	1 indicates that IPv6 support is configured 0 indicates that IPv6 support is not configured	1
supportlog	The location of the TimesTen daemon log file.	String	*timesten_home*/ diag/ttmesg.lo g.
userlog	The location of the TimesTen daemon user log file. The file location can be modified using the ttDaemonLog utility.	String	*timesten_ home*/diag/tter rors.log.
max_support_log_files	The TimesTen main daemon automatically rotates the files once they get to a specific size. This attribute specifies the maximum number of daemon log files to keep.	Integer	10
max_user_log_files	The TimesTen main daemon automatically rotates the files once they get to a specific size. This attribute specifies the maximum number of user log files to keep.	Integer	10
max_support_log_size_mb	The maximum size of a TimesTen daemon log file.	Integer	100 MB
max_user_log_size_mb	The maximum size of a TimesTen daemon user log file.	Integer	10 MB
daemon_log_snippet_ interval_in_mins	The duration of the daemon log collection for a critical event. The granularity of the duration is in minutes. The minimum value is 1 minute.	Integer	10

Attribute name	Description	Attribute value and type	Default
show_date	Indicates if the date should be prepended on all daemon and user log entries. 1 indicates that the date should be prepended to every record in the user and daemon log files. 0 indicates that the date is not prepended to every record in the user and daemon log files.	1 or 0	1
server_port	The port number on which the TimesTen server for this instance listens if you want the instance to be associated with a server.	A valid TCP port number.	6625
facility	If daemon logs are sent to syslog, the facility to be used. Possible name values are: auth, cron, daemon, local0-local7, lpr, mail, news, user, or uucp. The value is specified by the ttDaemonlog utility	String	None
server_stack_size	The client server stack size. The value is set by and can be changed by the ServerStackSize connection attribute.	Integer	768KB
servers_per_dsn	The number of servers per DSN for client server implementations. The value is set by and can be changed by the ServersPerDSN connection attribute	Integer	1
max_conns_per_server	The maximum number of client server connections for each TimesTen server. The value set by and can be changed by the MaxConnsPerServer connection attribute	Integer	1
server_pool	The number of processes that the TimesTen server should prespawn and keep in a reserve pool. If not specified, no processes are prespawned. The value is set by the MaxConnsPerServer connection attribute	Integer indicating the number of processes to prespawn	None

Attribute name	Description	Attribute value and type	Default
allow_network_files	Indicates if data access on NFS-mounted systems is allowed. By default, TimesTen systems cannot access data across NFS-mounted systems. On Linux x86 and Solaris, you can access checkpoint and transaction log files on NFS-mounted systems. 1 indicates data access on NFS-mounted systems is allowed. 0 indicates that access on NFS-mounted systems is not allowed. For more information, see "Using NFS-mounted systems for checkpoint and transaction log files" in the *Oracle TimesTen In-Memory Database Operations Guide*. TimesTen ignores this attribute on platforms other than Linux x86 and Solaris.	1 or 0	0
server_shmipc	Indicates if the TimesTen Server should use shared memory IPC. 1 indicates that the server uses shared memory IPC. 0 indicates that the server does not use shared memory IPC. For more information, see "Using shared memory for Client/Server IPC" in the *Oracle TimesTen In-Memory Database Operations Guide*.	1 or 0	1
server_shmsize	The size of the shared memory IPC.	Integer	64 MB
min_subs	The minimum number of subdaemons TimesTen spawns. The main TimesTen daemon spawns subdaemons dynamically as they are needed.	Integer	4
max_subs	The maximum number of subdaemons TimesTen spawns. The main TimesTen daemon spawns subdaemons dynamically as they are needed.	Integer	50

Attribute name	Description	Attribute value and type	Default
enable_policy_inactive	1 indicates that if there are too many failures to automatically load a database, TimesTen puts the database into a "policy inactive" mode. This is the default. 0 indicates that the RAM policy is one of Always, Manual or InUse. For more information, see "Specifying a RAM policy" in the *Oracle TimesTen In-Memory Database Operations Guide*.	1 or 0	1
noserverlog	Turns off logging of connects and disconnects from the client applications. 1 indicates logging is on. 0 indicates that logging is off.	1 or 0	1
replication_cipher_suite	The cipher suite to be used in encrypting communications to and from the replication agent. This setting is required if you are using TLS for replication.	String	None
replication_wallet	Specifies the path to the wallet directory—the directory where you placed the certificates that you generated. This setting is required if you are using TLS for replication. It is suggested, but not required, to use the same location and directory name on each TimesTen instance	Pathname	None
replication_ssl_mandatory	Specifies whether it is mandatory to have consistent TLS configuration between TimesTen instances—specifically, whether TLS is configured through replication_cipher_suite and replication_wallet settings, and what cipher suite is specified. If there is a mismatch between the current instance and the replication peer, then TimesTen behavior is determined as follows 1 indicates that replication cannot proceed unless settings are the same on all instances. 0 indicates that TLS is not used for communication between replication agents.	1 or 0	1

Attribute name	Description	Attribute value and type	Default
noserverlog	Turns off logging of connects and disconnects from the client applications. 1 indicates logging is on. 0 indicates that logging is off.	1 or 0	1
client_cipher_suites and server_cipher_suites	Lists the cipher suite or suites that can be used. These settings are required if you are using TLS for client server connections. You must set both settings. Set client_cipher_suites in the client timesten.conf file. Set server_cipher_suites in the server timesten.conf file For TLS to be used, the server and client settings must include at least one common suite. For more information, see "Transport Layer Security for TimesTen Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide*.	String	None
server_encryption	Specifies whether encryption is required or not for a client server connection. For more information, see "Transport Layer Security for TimesTen Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide*.	String	Accepted
client_wallet and server_wallet	Specifies the path to the wallet directory—the directory where you placed the certificates that you generated. These settings are required if you are using TLS for client server connections. Both attributes must be set. For more information, see "Transport Layer Security for TimesTen Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide*.	Pathname	None

Attribute name	Description	Attribute value and type	Default
ssl_client_authentication	Specifies whether TLS client authentication is required (setting of 1) or not (setting of 0, the default). With client authentication, the server validates an identity presented by the client, and requires an identity (public/private key) in the client wallet. Regardless of the client authentication setting, server authentication is performed, where the client validates the server. For more information, see Transport Layer Security for TimesTen Client/Server in the *Oracle TimesTen In-Memory Database Security Guide*.	1 or 0	1

2

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

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

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

- Required privileges for attributes
- List of attributes

Required privileges for attributes

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

List of attributes

This section includes the tables:

Table 2–1   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
Durability	Determines the durability of "prepare-to-commit" and commit records	If K-safety is set to 2, the default value is 0. If K-safety is set to 1, the default value is 1.
LogDir	The directory where transaction log files are stored.	None
Preallocate	Specifies that disk space for the checkpoint files should be preallocated when creating the database.	1 (True)
ReplicationApplyOrdering	Enables automatic parallel replication.	0 (Starts automatic parallel replication)
ReplicationParallelism	Specifies the number of tracks available for automatic parallel replication.	4

Table 2–1   (Cont.) Data store attributes

Name	Description	Default
Temporary	Specifies that the database is not saved to disk.	`0` (False)

Table 2–2   First connection attributes

Name	Description	Default
AutoCreate	Specifies that the first connection creates the database if it does not exist.	`1` (True)
BackupFailThreshold	Controls the number of log files that can accumulate before a backup fails.	`0` (Log files can continue to accumulate without a limit)
CkptFrequency	Controls the frequency in seconds that TimesTen performs a background checkpoint.	`0` (No predefined frequency)
CkptLogVolume	Controls the amount of data in megabytes that collects in the log between background checkpoints.	The value of the `LogFileSize` attribute
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 `SEMMNS` kernel parameter minus 155.
EpochInterval	Indicates the number of seconds between epochs.	If Durability = 1, the default is 0. (System does not generate periodic epochs) If Durability = 0, the default is 1.
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)
ForceDisconnectEnabled	Enables forced disconnections through `ttAdmin -disconnect`.	`0` (Disabled)
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`

Table 2–2   (Cont.) First connection attributes

Name	Description	Default
LogBufParallelism	The number of log buffer strands.	4
LogFileSize	The transaction log file size in MB.	The value of the `LogBufMB` attribute.
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
RecoveryThreads	The number of threads used to rebuild indexes during recovery.	4
TempSize	The size in MB for the temporary partition of the database.	The default size as determined from the `PermSize` value

Table 2–3   General connection attributes

Name	Description	Default
ChannelCreateTimeout	Specifies the time limit to wait for a channel create request to complete.	30,000 milliseconds
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.
CreateEpochAtCommit	Specifies if each commit generates an epoch.	0 (default) - TimesTen does not write the transaction log to disk on transaction commit.

Table 2–3   (Cont.) General connection attributes

Name	Description	Default
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`
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)
DurableCommits	Specifies that commit operations should write log records to disk.	`0` (Records not written to disk)
IncludeInCore	Specifies which parts of shared memory are included in a core dump.	0 (None)
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
OptimizerHint	Sets optimizer hints at the connection level.	None
PermWarnThreshold	The threshold at which TimesTen returns a warning 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 2-75.	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 if a query times out before executing.	`0` (No warning is returned)

Table 2–3   (Cont.) General connection attributes

Name	Description	Default
ReplicationTrack	Assigns a connection to a replication track.	None
SQLQueryTimeout	Specifies the time limit in seconds within which the database should execute SQL statements.	0 (No timeout)
SQLQueryTimeoutMSec	Specifies the time limit in milliseconds within which the database should execute SQL statements.	0 (No timeout)
TempWarnThreshold	The threshold at which TimesTen returns a warning when the temporary partition of the database is low in memory.	90 (percent)
UID See "UID and PWD" on page 2-75.	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 2–4   NLS general connection attributes

Name	Description	Default
ConnectionCharacterSet	The character encoding for the connection, which can be different from the database character set.	US7ASCII
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 2–5   PL/SQL first connection attributes

Name	Description	Default
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

Table 2–5   (Cont.) PL/SQL first connection attributes

Name	Description	Default
PLSQL_MEMORY_SIZE	The size in megabytes of the shared memory segment used by PL/SQL.	128 MB
PLSQL_OPEN_CURSORS	The number of open cursors.	50

Table 2–6   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_SESSION_CACHED_CURSORS	The number of session cursors to cache.	50
PLSQL_TIMEOUT	The number of seconds a PL/SQL procedure can run before being automatically terminated.	30

Table 2–7   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 2–8   TimesTen Cache database attributes

Name	Description	Default
CacheAWTParallelism	Indicates the number of threads necessary to apply changes to the Oracle database.	1
UseCacheConnPool	Enables or disables the cache connection pool.	0

Table 2–9   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)

Table 2–9   (Cont.) TimesTen Cache general connection attributes

Name	Description	Default
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
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)
StandbyNetServiceName	The Oracle Service Name of the standby Oracle database instance from which data is to be loaded into a TimesTen database. This attribute is only used by the cache agent in an Oracle Active Data Guard configuration. Set the `StandbyNetServiceName` attribute to the standby Oracle Service Name.	None

Table 2–10   TimesTen Client connection attributes

Name	Description	Default
CipherSuites	Lists the cipher suite or suites that can be used by TLS for client server connections, depending also on the client setting	None
ClientServer	Identify the client server process.	0
Encryption	Specifies whether encryption is required or not for a client server connection	`accepted`
SSLClientAuthentication	Specifies whether SSL client authentication is required	0
TCP_Port	The port number on which the TimesTen server is listening.	None
TCP_Port2, TCP_PortN	For TimesTen Classic, the port number on which the TimesTen server should listen if an automatic failover occurs.	None
TTC_ConnectTimeout	Number of seconds for the client to wait for a connect or disconnect call.	20 seconds
TTC_FailoverPortRange	A range for the failover port numbers.	None
TTC_NoReconnectOnFailover	Specifies if TimesTen performs an automatic reconnect after all other failover procedures are completed.	1

Table 2–10 (Cont.) TimesTen Client connection attributes

Name	Description	Default
TTC_Random_Selection	Specifies that the client randomly selects an alternate server from the list provided or selects the server according to `TTC_ServerN` settings.	0
TTC_REDIRECT	For TimesTen Scaleout, client/server connections, defines how a client is redirected.	1 (connect to any available server)
TTC_Redirect_Limit	For TimesTen Scaleout, limits the number of times the client is redirected.	1
TTC_RollbackRequiredOnFailover	Indicates if the application must roll back if a client failover occurs.	1 (enabled)
TTC_Server or TTC_Server1	Name of the computer where the TimesTen Server is running or a logical TimesTen server name.	None
TTC_Server2, TTC_ServerN	If an automatic failover occurs, the name of the system 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, TTC_Server_DSNn	For TimesTen Classic, server DSN corresponding to the TimesTen database, if an automatic failover occurs.	None
TTC_TCP_KEEPALIVE_INTVL_MS	The time interval (in milliseconds) between subsequential probes.	1000
TTC_TCP_KEEPALIVE_PROBES	The number of unacknowledged probes to send before considering the connection as failed and notifying the client.	2
TTC_TCP_KEEPALIVE_TIME_MS	The duration time (in milliseconds) between the last data packet sent and the first probe.	1000
TTC_Timeout	Timeout period, in seconds, for completion of a TimesTen client/server operation.	60
Wallet	The fully qualified path to the directory where the wallet is stored.	None

Table 2–11 TimesTen Server connection attributes

Name	Description	Default
CipherSuites	Lists the cipher suite or suites that can be used by TLS for client server connections, depending also on the client setting	None
Encryption	Specifies whether encryption is required or not for a client server connection	`accepted`
SSLClientAuthentication	Specifies whether SSL client authentication is required	0

Table 2–11 (Cont.) 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.	256
SSLRenegotiationPeriod	The time in minutes before which the client/server session is renegotiated.	0
SSLRenegotiationSize	The data transfer size in MB before which the client/server session is renegotiated.	0
Wallet	The fully qualified path to the directory where the wallet is stored.	None

Data store attributes

Data store attributes are set at database creation time. The data store attributes are listed in Table 2–1, " 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `Data Source Name` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	DSN	A name that describes the DSN.
Windows ODBC Data Source Administrator	Not applicable	

DataStore

The database path name uniquely identifies the physical database. It is the fully qualified directory path name of the database and the file name prefix, for example: /disk1/databases/database1. This name is not a file name. The actual database file names have suffixes, such as .ds0 and .log0, for example /disk1/databases/database1.ds0 and /disk1/databases/database1.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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set DataStore as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	DataStore	Full path to the physical database that the data source name references.
Windows ODBC Data Source Administrator	Not applicable	

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 and Linux systems (EUC) or Windows (SJIS)?

You should choose a connection character set that matches your terminal settings or data source. See "ConnectionCharacterSet" on page 2-79.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set DatabaseCharacterSet name as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	DatabaseCharacterSet	Specify the preferred character set.
Windows ODBC Data Source Administrator	Not applicable	

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
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

European character sets

Name	Description
BLT8CP921	Latvian Standard LVS8-92(1) Windows/UNIX/Linux 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

Name	Description
NE8ISO8859P10	ISO 8859-10 North European
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

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set Description as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	Description	Text description of the Data Source Name. This attribute is optional.
Windows ODBC Data Source Administrator	Not applicable	

Driver

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

For example, on Windows systems the value can be `TimesTen Client` 18.1.

On UNIX and Linux systems, the value of the `Driver` attribute is the path name of the TimesTen ODBC Driver shared library file, *timesten_home*/install/lib/libtten.so.

For more information, see "Creating a DSN on Linux and UNIX for TimesTen Classic" in *Oracle TimesTen In-Memory Database Operations Guide*.

For general usage scenarios, refer to standard ODBC reference documentation, such as

https://docs.microsoft.com/en-us/sql/odbc/reference/syntax/odbc-api-reference

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `Driver` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	Driver	Specifies the path name for the TimesTen ODBC Driver shared library file, if using a Driver Manager.
Windows ODBC Data Source Administrator	Select a driver from the **Create New Data Source** dialog.	Specifies the Client driver for TimesTen and the release.

Durability

For TimesTen Scaleout, the setting of this attribute determines the durability of "prepare-to-commit" and commit records.

If K-safety is set to 2, the default value is 0.

If K-safety is set to 1, the default value is 1.

`Durability=0` is not supported with K-safety set to 1.

For details about setting up K-safety, see "Creating a grid" in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is not supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `Durability` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems database definition (`.dbdef`) file in TimesTen Scaleout	Durability	0 - TimesTen does not write the transaction prepare-to-commit and commit records to disk on transaction commit.
		1 - TimesTen writes the prepare-to-commit record durably and the commit record non-durably.
Windows ODBC Data Source Administrator	N/A	

See also

CreateEpochAtCommit

LogDir

The LogDir attribute specifies the fully qualified directory path name 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. Placing the transaction log files and checkpoint files on different disks can improve system performance.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set LogDir as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	LogDir	Specifies the directory where transaction log files reside.
Windows ODBC Data Source Administrator	Not applicable	

Preallocate

The `Preallocate` attribute determines whether TimesTen preallocates disk space for the database checkpoint files 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.

TimesTen respects the setting for the `Preallocate` attribute for all operations that create a new checkpoint file, such as database creation, `ttRepAdmin -duplicate`, and `ttRestore`.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `Preallocate` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`Preallocate`	`0` - Does not preallocate disk space for checkpoint files when creating the database. `1` (Default) - Preallocates disk space for the checkpoint files.
Windows ODBC Data Source Administrator	Not applicable	

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

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.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `ReplicationApplyOrdering` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`ReplicationApplyOrdering`	`0` - Specifies automatic parallel replication. Automatic parallel replication is available for both classic and active standby pair replication schemes. (default)
		`2` - Specifies automatic parallel replication with disabled commit dependencies.
Windows ODBC Data Source Administrator	Not applicable	

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 greater 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.

ReplicationParallelism

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

Set ReplicationParallelism to a number from 2 to 32. The number cannot exceed half the value of LogBufParallelism. This number indicates the number of transmitter threads on the source database and the number of receiver threads on the target database. However, if you are using single-threaded replication, set ReplicationParallelism to 1, the default.

To learn more about automatic 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `ReplicationParallelism` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`ReplicationParallelism`	*n* - A value between 1 and 32, indicating the number of tracks to replicate in parallel. The default is 1.
Windows ODBC Data Source Administrator	Not applicable	

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.

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. 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `Temporary` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`Temporary`	`0` (default) - Creates a permanent database. `1` - Creates a temporary database.
Windows ODBC Data Source Administrator	Not applicable	

First connection attributes

TimesTen sets first connection attributes when a database created by the instance administrator is loaded into memory and persist for the first connection and all subsequent connections until the last connection to this database is closed.

You can modify first connection attributes only when the TimesTen database is unloaded. Then the instance administrator reconnects with new values for the first connection attributes.

First connection attributes are listed in Table 2–2, " 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 can be rejected or the attribute value can 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 2-46.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set AutoCreate as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	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	Not applicable	

BackupFailThreshold

Controls the number of transaction log files that accumulate in the directory defined by the value of the LogDir attribute after a backup starts before TimesTen releases the hold on checkpoint operations. If a checkpoint is initiated before the completion of a backup, the backup is invalidated.

TimesTen temporarily ignores the CkptFrequency and CkptLogVolume attributes (controlling background checkpoints) while a backup is in progress if this attribute is not set or is set to 0.

Set the attribute to a value that is high enough to ensure the safe completion of your backup. For example, if a backup typically takes n seconds to complete and your database creates m transaction log files per second, set BackupFailthreshold to a value greater than $n*m$.

The number of log files generated by your database per any given unit of time is directly proportional to your write workload and inversely proportional to the value set for the LogFileSize attribute.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set BackupFailThreshold as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	BackupFailThreshold	Enter a non-zero integer value that indicates the number of transaction log files that are allowed to accumulate before the backup fails.
		The default is 0, indicating that transaction log files continue to accumulate with the backup operation is in process.
Windows ODBC Data Source Administrator	Not applicable	

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.

When using TimesTen Scaleout, if Durability = 0, set EpochInterval to a value less than the value of the `CkptFrequency` when `Durability=0` to guarantee at least 1 epoch per interval.

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.

When the value of this attribute is more than 0, if a checkpoint fails, TimesTen attempts a checkpoint once every 30 seconds. 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `CkptFrequency` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`CkptFrequency`	Enter a value in seconds for the frequency at which TimesTen should perform a background checkpoint. The default is 0.
		If you do not specify this attribute with a value, TimesTen does not perform background checkpoints. For an existing database, TimesTen uses the stored value.
Windows ODBC Data Source Administrator	Not applicable	

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.

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.

When the value of this attribute is more than `0` and `CkptFrequency=0`, if a checkpoint fails, TimesTen attempts a checkpoint every 30 seconds. 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `CkptLogVolume` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	CkptLogVolume	Specify the amount of data in megabytes that can accumulate in the transaction log file between background checkpoints. The default is the value supplied for the LogFileSize attribute.
		For an existing database, TimesTen uses the stored value.
		If the attribute is specified, but you do not supply a value, TimesTen uses the value supplied for the LogFileSize attribute.
Windows ODBC Data Source Administrator	Not applicable	

CkptRate

Controls the maximum rate at which data should be written to disk during a checkpoint operation. This can 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set CkptRate as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	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	Not applicable	

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 (in parallel). 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `CkptReadThreads` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	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	Not applicable	

Notes

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

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.

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.

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 2-89.

> **Note:** The kernel must be configured with enough semaphores to handle all active databases. For details on setting semaphores for your system, see "Operating system prerequisites" in *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide* or "Operating system prerequisites" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set Connections as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	Connections	The default value is the lesser of 2000 or the number of semaphores specified in the SEMMNS kernel parameter minus 155. A legal value is an integer from 1 through 32000. The value represents the maximum number of connections.
Windows ODBC Data Source Administrator	Not applicable	

EpochInterval

For TimesTen TimesTen Scaleout, use this attribute to set the number of seconds between epochs.

If Durability = 1, the default for this attribute is 0. (System does not generate periodic epochs)

If Durability = 0, the default for this attribute is 1. You must set this attribute to a value less than the value of the `CkptFrequency` when `Durability=0` to guarantee at least 1 epoch per interval.

If this attribute is set to `0`, the system does not generate periodic epochs. An application can generate epochs at custom intervals by calling the `ttEpochCreate` built-in procedure each time the application wants to create an epoch.

If an element is down, an epoch interval can be skipped.

As long as one element in each replica set is up, the system never skips more than `K*EpochInterval` seconds between epochs.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is not supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `EpochInterval` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems database definition (`.dbdef`) file in TimesTen Scaleout	`EpochInterval`	`0` - TimesTen does not generate periodic epochs. `n > 0` - An integer that indicates the number of seconds between epoch writes.
Windows ODBC Data Source Administrator	N/A	

See also

CreateEpochAtCommit

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set ForceConnect as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	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	Not applicable	

ForceDisconnectEnabled

Allows you to force disconnection from the database using `ttAdmin -disconnect`. See "Force disconnect" on page 5-11.

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

Notes:

- By default, the forced disconnect feature is disabled. Existing direct-connect applications may find it undesirable for TimesTen to spawn the thread that is required to implement this functionality.

- Users should not specify different values of this attribute for different database connections. If the force disconnect feature is desired, add `ForceDisconnectEnabled=1` to the DSN definition in the `sys.odbc.ini` file.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `ForceDisconnectEnabled` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`ForceDisconnectEnabled`	`0` (default) - Do not allow forced disconnections. `1` - Allow forced disconnections.
Windows ODBC Data Source Administrator	N/A	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set LogAutoTruncate as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	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	Not applicable	

LogBufMB

For TimesTen Classic,The `LogBufMB` attribute specifies the size of the internal transaction log buffer in megabytes, for the database.

For TimesTen Scaleout,The `LogBufMB` attribute specifies the size of the internal transaction log buffer in megabytes, for the element.

The recommendation is to set `LogBufMB` to a value between 256 MB and 4 GB. If memory space is a concern, start with 256 MB; otherwise, start with 1 GB.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `LogBufMB` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`LogBufMB`	*n* - Size of log buffer in megabytes.
		If not set and the database or element exists, TimesTen uses the value stored in the database.
		If not set and the database or element is being created, TimesTen uses the default value of `64` MB.
		The maximum value is 65,536 MB (64 GB).
Windows ODBC Data Source Administrator	Not applicable	

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. Strands divide the transaction log buffer available memory into a number of different regions, which can be accessed concurrently by different connections. Each connection can execute data-independent DML statements in parallel using those strands as if each has its own transaction log buffer.

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.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set LogBufParallelism as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	LogBufParallelism	An integer value between 1 and 64. Default is 4.
Windows ODBC Data Source Administrator	Not applicable	

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 can 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `LogFileSize` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`LogFileSize`	*n* - Size of transaction log file in megabytes. Default is the value of LogBufMB when the database is created and `0` (current size in effect) on subsequent connections. The minimum size is 8 MB. The maximum value is 65,536 MB (64 GB).
Windows ODBC Data Source Administrator	Not applicable	

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 1 if most of your transactions commit durably and use the value 0 otherwise.

For best results, however, experiment with both values using a typical workload for your application and platform. Although application performance can 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set LogFlushMethod as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	LogFlushMethod	0 (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).
		1 - Write data to transaction log files using synchronous writes such that explicit sync operations are not needed.
Windows ODBC Data Source Administrator	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `LogPurge` as follows:

Where to set the attributes	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	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	Not applicable	

MemoryLock

On Windows 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 Linux, `MemoryLock=4` will try to obtain a `MemoryLock` based on what the operating system allows. On Linux, locking all of the virtual memory size (physical + swap) can occur. TimesTen performs best if it does not use swap. Since the operating system allows locking more memory than is actually available, it is important to carefully configure the operating system memory management parameters to limit the amount of lockable memory. You can configure these parameters in the `/etc/security/limits.conf` file.

On AIX the `MemoryLock` attribute is not implemented.

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.

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
```

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `MemoryLock` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`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	Not applicable	

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 2-27). 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `Overwrite` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`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	N/A	

PermSize

For TimesTen Classic, indicates the size in MB of the permanent memory region for the database.

For TimesTen Scaleout, indicates the size in MB of the permanent memory region for the element.

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 memory region sizes 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `PermSize` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`PermSize`	*n* - Size of permanent partition in megabytes; default is 32 MB. Minimum size is 32 MB.
Windows ODBC Data Source Administrator	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `RecoveryThreads` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`RecoveryThreads`	n - The number of threads to use when rebuilding indexes during recovery. Default is 4 when the database is created and 0 on subsequent connections.
Windows ODBC Data Source Administrator	Not applicable	

Notes

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

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

TempSize

For TimesTen Classic, `TempSize` indicates the total amount of memory in MB allocated to the temporary region for the database

For TimesTen Scaleout, indicates the total amount of memory in MB allocated to the temporary region for an element

`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 can 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TempSize` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TempSize`	*n* - Size of the temporary partition, in MB. Minimum size is 32 MB on all platforms.
Windows ODBC Data Source Administrator	Not applicable	

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 2–3, " General connection attributes" and described in detail in this section.

ChannelCreateTimeout

Use this attribute to determine the time limit in milliseconds to wait for a response to a channel create request. The value used for the channel create timeout is determined by comparing the values set for the `ChannelCreateTimeout`, `SQLQueryTimeoutMSec` and `SQLQueryTimeout` connection attributes.

- If `ChannelCreateTimeout`, `SQLQueryTimeout` and `SQLQueryTimeoutMSec` are non-zero, then the timeout used is the minimum value of these connection attributes.

- If `ChannelCreateTimeout` is set to `0`, then irrespective of the value of `SQLQueryTimeout`, channel timeout feature will be disabled and channel creation will be synchronous.

- If `ChannelCreateTimeout` is set to a non-zero value and `SQLQueryTimeout` or `SQLQueryTimeoutMSec` are all set to `0`, then the value of `ChannelCreateTimeout` is the timeout for channel creation.

> **Note:** If the timeout used is different than what is configured in the `ChannelCreateTimeout` connection attribute, TimesTen does not overwrite the value set for the `ChannelCreateTimeout` connection attribute.

See "Choose SQL and PL/SQL timeout values" in the *Oracle TimesTen In-Memory Database Operations Guide* for more information.

The timeout for channel create should not exceed the value specified for `TTC_Timeout`, which is a timeout for when the TimesTen Client application waits for a result from the corresponding TimesTen Server process. See "TTC_Timeout" on page 2-138 for details on this connection attribute.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is not supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `ChannelCreateTimeout` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems database definition (`.dbdef`) file in TimesTen Scaleout	`ChannelCreateTimeout`	*n* - Time limit in milliseconds for how long to wait on a create channel request. The value of *n* can be any integer equal to or greater than 0. The default value is 30,000 milliseconds (30 seconds). If you set the value to `0`, then the query does not time out.

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `CommitBufferSizeMax` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`CommitBufferSizeMax`	`0` - Commit buffer is configured to the default size. (10MB). *n* - Integer value. Minimum user configured size is 1 (MB), Configure this value to a value much smaller than TempSize.
Windows ODBC Data Source Administrator	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set ConnectionName as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	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$n$, where n is an integer greater than 0 to make the name unique. If not specified, the connecting process name.
Windows ODBC Data Source Administrator	**Connection** field on the Oracle TimesTen Client DSN Setup dialog	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$n$, where n is an integer greater than 0 to make the name unique. If not specified, the connecting process name.

CreateEpochAtCommit

For TimesTen Scaleout, specifies if each commit generates an epoch.

Set this general connection attribute to '1' to make every commit from this connection an epoch. Default value is '0' which will make it so commits are not epochs unless the transaction included a call to the `ttEpochCreate` built-in procedure.'

TimesTen Classic ignores this attribute.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is not supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `CreateEpochatCommit` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems database definition (`.dbdef`) file in TimesTen Scaleout	`CreateEpochatCommit`	`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	Not applicable	

See also

LogFlushMethod

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

The value can 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*.

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 "Making DDL changes in an active standby pair" in the *Oracle TimesTen In-Memory Database Replication Guide*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `DDLReplicationAction` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`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	Not applicable	

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 INDEX` or `DROP INDEX`

- `CREATE SYNONYM` or `DROP SYNONYM`

- `CREATE TABLE` 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 VIEW` or `DROP VIEW`

- `CREATE SEQUENCE` 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 and passwords" in the *Oracle TimesTen Application-Tier Database Cache User's Guide* or "ttCacheUidPwdSet" on page 3-83.

The value of this attribute can 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 "Making DDL changes in an active standby pair" in the *Oracle TimesTen In-Memory Database Replication Guide*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set DDLReplicationLevel as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	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	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `Diagnostics` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	Diagnostics	`0` - TimesTen does not generate diagnostic messages. `1` (default) - TimesTen generates base-level diagnostics messages.
Windows ODBC Data Source Administrator	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set DurableCommits as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	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	Not applicable	

See also

LogFlushMethod

IncludeInCore

This attribute allows the application to control whether TimesTen shared memory should be included in application core dumps, and which portions of that memory should be included.

If multiple connections exist from a single application process to a single TimesTen database, the `IncludeInCore` value of the most recent connection of the process determines the parts of the core file to dump.

For Client/Server connections, the setting is passed to TimesTen Server, which passes it on to the direct driver.

TimesTen daemons always dump everything.

The settings noted below are additive. For example, set `IncludeInCore` to 3 (1+2) for DB header and other fixed allocations plus perm space. Set it to 15 (1+2+4+8) for DB header and other fixed allocations plus perm space, temp space, and log buffer.

Required privilege

`ADMIN` privilege is required to include the DB header and other fixed allocations.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `IncludeInCore` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	IncludeInCore	0 - Exclude the shared memory associated with this database connection from any core files.
		1 - Include the DB header and other fixed allocations from this database in core files. This is necessary to make sense of other information you request.
		2 - Include perm space from this database in core files.
		4 - Include temp space from this database in core files.
		8 - Include the log buffer from this database in core files.
		16 - Include the PL/SQL shared memory from this database in core files.
		The default value is 255.
Windows ODBC Data Source Administrator	N/A	

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 can 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 2-110.

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_181:

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

```
% ttIsql;
Command> connect "dsn=repdb1_181;passthrough=3";
Connection successful:. . .PassThrough=3;
<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;
        commit;
        exec=1;
```

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout. Only Isolation=1 is supported in TimesTen Scaleout.

Setting

Set Isolation as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`Isolation`	`0` - Connects to database in serializable isolation mode. `1` (default) - Connects to database in read committed mode.
Windows ODBC Data Source Administrator	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set LockLevel as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	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	Not applicable	

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` can 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 can 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.

A connection can change the lock wait interval at any time by calling the 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `LockWait` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	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	Not applicable	

OptimizerHint

The `OptimizerHint` connection attribute establishes the connection's optimizer hint defaults. They can be different than the system defaults. The optimizer hints set with this connection attribute are set for every SQL statement in the user application.

The value of this attribute is a string of the same format as the statement level optimizer hints, but without the delimiters `*+`, `*/` and `-+`. The string can only contain the optimizer hint names. It cannot be mixed with other hint strings or comments.

The order of precedence for optimizer hints is statement level hints, transaction level hints and lastly hints set by this connection attribute.

For client/server applications, the attribute set by the client connection takes precedence over server DSN settings of this attribute.

Some symbols, such as semi-colons (`;`) are not accepted in attribute values. For hints where the parameter might contain a semi-colon, multiple hints of the same name are combined into one hint. For example, to express:

```
TT_INDEX (t1,i1, 0; t2, i2,0)
```

use

```
TT_INDEX (t1,i1,0) TT_INDEX (t2, i2,0 )
```

To combine multiple hints at the connection level, you must enter them in the same line.

For a list of optimizer hints supported as values to this attribute, 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `OptimizerHint` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	OptimizerHint	A string specifying optimizer hints and their values. The maximum length of the string is 512.
Windows ODBC Data Source Administrator	Not applicable	

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.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `PermWarnThreshold` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`PermWarnThreshold`	*p* - Percentage at which TimesTen should issue out-of-memory warnings. Default is `90`.
Windows ODBC Data Source Administrator	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set PrivateCommands as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	PrivateCommands	0 (default) - Commands are shared with other connections. 1 - Commands are not shared with any other connection.
Windows ODBC Data Source Administrator	Not applicable	

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 Linux systems 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 2-75.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set PWDCrypt as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	PWDCrypt	Enter the value generated by the ttUser utility.
Windows ODBC Data Source Administrator	**PWDCrypt** field on the Oracle TimesTen Client DSN Setup dialog	Enter the value generated by the ttUser utility.

QueryThreshold

Use this attribute to write a warning to the daemon log when the execution time of a SQL statement exceeds the specified value. 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set QueryThreshold as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	QueryThreshold	A nonnegative integer. Default is 0 and indicates that TimesTen does not return a warning.
Windows ODBC Data Source Administrator	Not applicable	

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 to 2.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `ReplicationTrack` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	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	Not applicable	

SQLQueryTimeout

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

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.

Both SQLQueryTimeout and SQLQueryTimeoutMSec attributes are internally mapped to one timeout value in milliseconds. If different values are specified for these attributes, only one value is retained.

For more details, see "Choose SQL and PL/SQL timeout values" in the *Oracle TimesTen In-Memory Database Operations Guide*.

> **Note:**
>
> - When SQL query timeouts are used (SQLQueryTimeout or SQLQueryTimeoutMsec), TimesTen behavior is on a best-effort basis. It is not possible to guarantee that the timeout will actually occur within the specified time.
>
> - SQL query timeouts are honored during dynamic load unless the dynamic load requires a new connection to the Oracle database, in which case the connection is allowed to complete. If the connection completes successfully but the dynamic load times out, the connection will be retained.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set SQLQueryTimeout as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	SQLQueryTimeout	*n* - Time limit in seconds for which the database should execute SQL queries. The value of *n* 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.
Windows ODBC Data Source Administrator	Not applicable	

SQLQueryTimeoutMSec

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

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.

Both SQLQueryTimeout and `SQLQueryTimeoutMsec` attributes are internally mapped to one timeout value in milliseconds. If different values are specified for these attributes, only one value is retained.

For more details, see "Choose SQL and PL/SQL timeout values" in the *Oracle TimesTen In-Memory Database Operations Guide*.

> **Notes:**
>
> - When SQL query timeouts are used (`SQLQueryTimeout` or `SQLQueryTimeoutMsec`), TimesTen behavior is on a best-effort basis. It is not possible to guarantee that the timeout will actually occur within the specified time.
>
> - SQL query timeouts are honored during dynamic load unless the dynamic load requires a new connection to the Oracle database, in which case the connection is allowed to complete. If the connection completes successfully but the dynamic load times out, the connection will be retained.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `SQLQueryTimeoutMsec` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`SQLQueryTimeoutMSec`	n - Time limit in milliseconds for which the database should execute SQL queries. The value of n 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.
Windows ODBC Data Source Administrator	Not applicable	

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. See "ttWarnOnLowMemory" on page 3-267.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TempWarnThreshold` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	TempWarnThreshold	*p* - Percentage at which warning should be issued. Default is 90.
Windows ODBC Data Source Administrator	Not applicable	

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.

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 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.

Usage in TimesTen Scaleout and TimesTen Classic

These attributes are supported in TimesTen Classic.

These attributes are supported in TimesTen Scaleout.

Setting

Set `UID` and `PWD` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	UID	Character string specifying the user ID.
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	PWD	Character string specifying the password that corresponds to the user ID.
Windows ODBC Data Source Administrator	**User ID** and **Password** fields on the Oracle TimesTen Client DSN Setup dialog	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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `WaitForConnect` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	WaitForConnect	0 - Does not wait if connection to database fails. 1 (default) - Waits until connection to database is possible.
Windows ODBC Data Source Administrator	Not applicable	

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 2–4, " 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 can 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 or Linux (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.

This attribute accepts the same values used for the DatabaseCharacterSet. For a list of supported character set names, see "Supported character sets" on page 2-14.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set ConnectionCharacterSet as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	ConnectionCharacterSet	The default value for ConnectionCharacterSet is US7ASCII.
Windows ODBC Data Source Administrator	**Connection CharacterSet** list field on the Oracle TimesTen Client DSN Setup dialog	The default value for ConnectionCharacterSet is US7ASCII.

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.

NLS_LENGTH_SEMANTICS can be modified by an ALTER SESSION SQL statement, described in *Oracle TimesTen In-Memory Database SQL Reference*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set NLS_LENGTH_SEMANTICS as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	NLS_LENGTH_ SEMANTICS	Specify either BYTE (default) or CHAR.
Windows ODBC Data Source Administrator	Not applicable	

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.

NLS_NCHAR_CONV_EXCP can be modified by an ALTER SESSION SQL statement, described in *Oracle TimesTen In-Memory Database SQL Reference*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

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 and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	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	Not applicable	

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

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

NLS_SORT can be modified by an ALTER SESSION SQL statement, described in *Oracle TimesTen In-Memory Database SQL Reference*.

> **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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set NLS_SORT as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	NLS_SORT	Specify the linguistic sort sequence or BINARY (default).
Windows ODBC Data Source Administrator	Not applicable	

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

Basic name	Extended name
GBK	-
GREEK	-
HEBREW	-
HKSCS	-
HUNGARIAN	XHUNGARIAN
ICELANDIC	-
INDONESIAN	-
ITALIAN	-
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.

Sort name	Description
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.
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 2–5, " PL/SQL first connection attributes" and described in detail in this section.

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 you do not specify 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. -64

The platform specific default memory addresses are:

Operating system	Address
Linux x86-64	0000005000000000
AIX	06ffffff00000000
Windows	000000005b8c0000
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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `PLSQL_MEMORY_ADDRESS` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	PLSQL_MEMORY_ADDRESS	A hexidecimal value that indicates the memory address for PL/SQL process.
Windows ODBC Data Source Administrator	Not applicable	

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.

- 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.

- 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 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set PLSQL_MEMORY_SIZE as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	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 128 MB.
Windows ODBC Data Source Administrator	Not applicable	

PLSQL_OPEN_CURSORS

This attribute specifies the maximum number of PL/SQL cursors that can be open in a session at one time. Use this to prevent a session from opening an excessive number of cursors. Default is 50 PL/SQL cursors.

Updating the value of this attribute takes effect on the next connection, not the current connection.

If you decrease the value and the number of open cursors currently exceeds or equals the new setting, no new cursors can be opened until the total number of open cursors is less than the new setting (i.e., some of the currently open cursors have to close).

A value of 0 indicates no PL/SQL cursors can be open. (But if there are cached PL/SQL cursors that contain any PL/SQL code, they could still be executed.)

> **Note:**   This attribute has the same functionality as OPEN_CURSORS in Oracle Database.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set PLSQL_OPEN_CURSORS as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	PLSQL_OPEN_CURSORS	A positive integer from 0 to 65535 representing the number of cursors that can be open in one session at one time. The default value is 50.
Windows ODBC Data Source Administrator	N/A	

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 2–6, " 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. Possible values are IDENTIFIERS:NONE or IDENTIFIERS:ALL.

The PLSCOPE_SETTINGS connection attribute determines the initial value of this attribute within a session. The value can be modified by an ALTER SESSION statement, described in *Oracle TimesTen In-Memory Database SQL Reference*. For example:

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

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set PLSCOPE_SETTINGS as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	PLSCOPE_SETTINGS	IDENTIFIERS:NONE (default) IDENTIFIERS:ALL
Windows ODBC Data Source Administrator	Not applicable	

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 are 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.

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';
```

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set PLSQL_CCFLAGS as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	PLSQL_CCFLAGS	'A string literal with this format: '*v1:c1,v2:c2,...,vn:cn*' Default: null
Windows ODBC Data Source Administrator	Not applicable	

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;
  ```

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

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 and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	PLSQL_CONN_MEM_LIMIT	An integer value in MB. Default value is 100. A setting of 0 means no limit.
Windows ODBC Data Source Administrator	Not applicable	

Notes

The amount of space consumed by PL/SQL variables is roughly what you might expect comparable variables to consume in other programming languages. As an example, consider a large array of strings:

```
type chararr is table of varchar2(32767)
             index by binary_integer;
big_array chararr;
```

If 100,000 strings of 100 bytes each are placed into such an array, approximately 12 megabytes of memory is consumed.

Memory consumed by variables in PL/SQL blocks is used while the block executes, then is released. Memory consumed by variables in PL/SQL package specifications or

bodies (not within a procedure or function) is used for the lifetime of the package. Memory consumed by variables in a PL/SQL procedure or function, including one defined within a package, is used for the lifetime of the procedure or function. However, in all cases, memory freed by PL/SQL is not returned to the operating system. Instead, it is kept by PL/SQL and reused by future PL/SQL invocations. The memory is freed when the application disconnects from TimesTen.

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. Possible values are 0, 1, 2, or 3.

The PLSQL_OPTIMIZE_LEVEL connection attribute determines the initial value of this attribute within a session. The value can be modified by an ALTER SESSION statement, described in *Oracle TimesTen In-Memory Database SQL Reference*. For example:

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

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set PLSQL_OPTIMIZE_LEVEL as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	PLSQL_OPTIMIZE_LEVEL	The default value is 2.
Windows ODBC Data Source Administrator	Not applicable	

PLSQL_SESSION_CACHED_CURSORS

This attribute specifies the number of session cursors to cache. A user may adjust the setting to free up space not currently needed in the cache.

PLSQL_SESSION_CACHED_CURSORS can be modified by an ALTER SESSION SQL statement, described in *Oracle TimesTen In-Memory Database SQL Reference*.

ALTER SESSION SET PLSQL_SESSION_CACHED_CURSORS=25;

> **Note:** This attribute has the same functionality as SESSION_CACHED_ CURSORS in Oracle Database.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set PLSQL_SESSION_CACHED_CURSORS as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	PLSQL_SESSION_CACHED_ CURSORS	A positive integer from 1 to 65535 representing the number of cursors to cache. The default value is 50.
Windows ODBC Data Source Administrator	N/A	

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 can 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:**
>
> - See "Choose SQL and PL/SQL timeout values" in *Oracle TimesTen In-Memory Database Operations Guide* for information about the relationship between TTC_Timeout, SQLQueryTimeout, and PLSQL_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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set PLSQL_TIMEOUT as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	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	Not applicable	

See also

TTC_TCP_KEEPALIVE_TIME_MS

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 2–7, " 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.

> **Notes:**
>
> - This attribute can also be set through the `ttDBConfig` built-in procedure, which overrides the connection attribute setting. See "ttDBConfig" on page 3-111.
>
> - 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `CacheAWTMethod` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`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	Not applicable	

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.

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 to the maximum value of 31.

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.

CacheAWTParallelism only has an affect when there are AWT cache groups.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set CacheAWTParallelism as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	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	Not applicable	

UseCacheConnPool

Enable the cache connection pool with the `UseCacheConnPool` connection attribute.

The cache connection pool can only be initiated from client-server applications and is used only for dynamic loads initiated for dynamic read-only cache groups.

To learn more about the cache connection pool, see "Managing a cache connection pool to the Oracle database for dynamic load requests" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `UseCacheConnPool` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`UseCacheConnPool`	`2` - Enabled: For each statement that requires a dynamic load from the Oracle database, the connections to Oracle are released after the load completes and the connection is returned to the cache connection pool.
		If a dynamic load is performed from a direct connection, the dynamic load proceeds as if the feature is not enabled.
		`0` (default) - Disabled: For each statement that requires a dynamic load from the Oracle database, the connections to Oracle are not closed by TimesTen after the load completes.
Windows ODBC Data Source Administrator	N/A	

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 2–9, " 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `DynamicLoadEnable` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`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	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `DynamicLoadErrorMode` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`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	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `OracleNetServiceName` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`OracleNetServiceName`	Character string specifying the Oracle Service Name that is to be used as the Oracle ID.
Windows ODBC Data Source Administrator	Not applicable	

OraclePWD

The TimesTen Cache uses the `OraclePWD` attribute to connect to the Oracle database to perform cache operations.

The Oracle database user has the same user name as the TimesTen cache manager user, whose user name is specified by `UID` (and whose password on TimesTen is specified by `PWD`). Hence, the value of this attribute is the password for the user specified by `UID`

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

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;
(Default setting AutoCommit=1)
```

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 2-62.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `PassThrough` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`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.
Windows ODBC Data Source Administrator	Not applicable	

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `RACCallback` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`RACCallback`	`0` - Do not install TAF and FAN callbacks.
		`1` (default) - Install the TAF and FAN callbacks.
Windows ODBC Data Source Administrator	Not applicable	

StandbyNetServiceName

The TimesTen Cache uses the `StandbyNetServiceName` attribute.

This attribute identifies the Service Name for the standby Oracle instance from an Oracle Active Data Guard environment.

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

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported with TimesTen Scaleout.

Setting

Set `StandbyNetServiceName` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	StandbyNetServiceName	Character string specifying the Oracle Service Name that is to be used as the standby Oracle ID.
Windows ODBC Data Source Administrator	Not applicable	

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 2–10, " 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

- PWDCrypt

To view the value of a client attribute:

- In ODBC 3.5, use the ODBC function SQLGetConnectAttr. To learn more about this function, see "Attribute support for ODBC 3.5 SQLSetConnectAttr and SQLGetConnectAttr" in *Oracle TimesTen In-Memory Database C Developer's Guide*

- In ODBC 2.5, use the ODBC function SQLGetConnectOption. To learn more about this function, see "Option support for ODBC 2.5 SQLSetConnectOption and SQLGetConnectOption" section of the *Oracle TimesTen In-Memory Database C Developer's Guide*.

CipherSuites

The CipherSuites attribute lists the cipher suite or suites that can be used, depending also on the client setting. Specify SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, or both, comma-separated and in order of preference. There is no default setting.

You must set this attribute for both the client and the server. For TLS to be used, the server and client settings must include at least one common suite.

See "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set CipherSuites as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	CipherSuites	Specify SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, or both, comma-separated and in order of preference
Windows ODBC Data Source Administrator	**CipherSuites** field on the Oracle TimesTen Client DSN Setup dialog.	Specify SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, or both, comma-separated and in order of preference

ClientServer

Identify the client server process with the `ClientServer` connection attribute.

In a client/server environment, TimesTen can have multiple TimesTen child server processes to handle incoming requests from clients. You provide the `ChildServer` connection attribute to identify a specific child server process for certain cache connection pool built-in procedures.

Each child server process is identified by a number assigned with the `ChildServer=`*n* connection attribute, where *n* is a number ranging from 1 to the number of running child server processes. Once connected to the child server process, you can execute either the `ttCacheConnPoolGet('current')` or `ttCacheConnPoolApply` built-in procedures that are meant for a specific child server process.

To learn more about the `ClientServer` connection attribute, see "Managing a cache connection pool to the Oracle database for dynamic load requests" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `ClientServer` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`ClientServer`	`ChildServer=`*n*, where *n* is a number ranging from 1 to the number of running child server processes. The default value for the `ChildServer` connection attribute is 0.
Windows ODBC Data Source Administrator	Not applicable	

Encryption

The Encryption attribute specifies whether encryption is required or not for a client server connection.

You must set this attribute for both the client and the server. Cipher settings must be the same on both the client and server, in most cases.

See "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set Encryption as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	Encryption	■ accepted: Enable an encrypted session if required or requested by the client; use an unencrypted session otherwise. (default) ■ rejected: Demand an unencrypted session. (If the server does not support encryption, TimesTen behaves as if this is the setting on the server.) The connection is rejected if the client requires encryption. ■ requested: Request an encrypted session if the client allows it (if the client has any setting other than rejected); use an unencrypted session otherwise. ■ required: Demand an encrypted session. Reject the connection if the client rejects encryption.
Windows ODBC Data Source Administrator	**Encryption** field on the Oracle TimesTen Client DSN Setup dialog.	■ accepted: Enable an encrypted session if required or requested by the client; use an unencrypted session otherwise. (default) ■ rejected: Demand an unencrypted session. (If the server does not support encryption, TimesTen behaves as if this is the setting on the server.) The connection is rejected if the client requires encryption. ■ requested: Request an encrypted session if the client allows it (if the client has any setting other than rejected); use an unencrypted session otherwise. ■ required: Demand an encrypted session. Reject the connection if the client rejects encryption.

SSLClientAuthentication

The `SSLClientAuthentication` attribute specifies whether SSL client authentication is required (setting of 1) or not (setting of 0, the default). With client authentication, the server validates an identity presented by the client, and requires an identity (public/private key) in the client wallet.

Regardless of the client authentication setting, server authentication is performed, where the client validates the server.

You must set this attribute for both the client and the server. Regardless of the client authentication setting, server authentication is performed, where the client validates the server.

See "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `SSLClientAuthentication` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`SSLClientAuthentication`	1 - Client authentication is required 0 - No authentication is required. (the default)
Windows ODBC Data Source Administrator	**SSL Client Authentication** field on the Oracle TimesTen Client DSN Setup dialog.	1 - Client authentication is required 0 - No authentication is required. (the default)

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.

The default TCP/IP port number is assumed for TCP_Port unless you specify a value in the TTC_Server connection attribute, in the ODBC connection string, or in the logical server definition. See "TTC_Server or TTC_Server1" on page 2-129 for more details.

If the TimesTen Server is listening on a non-default port number, you must provide the port number in one of the following ways:

- If using TimesTen Classic, you can specify the port number within the logical server definition, which contains the network address and port number pair. See "Defining a logical server name" in *Oracle TimesTen In-Memory Database Operations Guide* for more information on defining a logical server.

- You can specify the port number within the TTC_Server connection attribute using:

  ```
  TTC_SERVER=server_host_name/server_port;
  ```

- You can specify the port number in the ODBC connection string.

  ```
  "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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

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 and Linux systems ttconnect.ini file. See Creating and configuring a logical server name on Linux and UNIX in the *Oracle TimesTen In-Memory Database Operations Guide* for more details.	**TCP_Port** on the Oracle TimesTen Logical Server Name Setup dialog.	Specify the port number where the Server is listening.

TCP_Port2, TCP_PortN

For TimesTen Classic, TimesTen uses this attribute to specify the port number to use if an automatic failover occurs. (This is unnecessary for TimesTen Scaleout.) See the description of TCP_Port for details on setting the value of this attribute and associated attributes.

The default TCP/IP port number is assumed for `TCP_Port2` and `TCP_PortN` unless you specify a value in the appropriate `TTC_ServerN` connection attribute, in the ODBC connection string, or in the logical server definition. See "TTC_Server or TTC_Server1" on page 2-129 for more details.

Unspecified values for `TCP_PortN` inherit the value of `TCP_PORT` (or `TCP_PORT1`). For example, if `TTC_Server2` is specified but `TTC_Server_DSN2` and `TCP_Port2` are not, then `TTC_Server_DSN2` is set to the `TTC_Server_DSN` value and `TCP_Port2` is set to the `TCP_Port` value.

See "Using automatic client failover" in *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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TCP_Port2, TCP_PortN` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs	`TCP_Port2, TCP_PortN`	Specify the failover port number where the Server should listen.
Windows ODBC Data Source Administrator and UNIX and Linux systems `ttconnect.ini` file. See Creating and configuring a logical server name on Linux and UNIX in the *Oracle TimesTen In-Memory Database Operations Guide* for more details.	Specify **TCP_Port2, TCP_PortN** in a connection string when also specifying a TTC_Server2, TTC_ServerN.	Specify the failover port number where the Server should listen.

TTC_ConnectTimeout

The `TTC_ConnectTimeout` attribute specifies the maximum number of seconds the client waits for a `SQLDriverConnect` or `SQLDisconnect` request. It overrides the value of `TTC_Timeout` for those requests. Set the `TTC_ConnectTimeout` when you want connection requests to timeout with a different timeframe than the timeout provided for query requests. For example, you can set a longer timeout for connections if you know that it takes a longer time to connect, but set a shorter timeout for all other queries.

A value of 0 means there is no timeout. A negative value defers to the `TTC_Timeout` setting. As with `TTC_Timeout`, if the timeout is reached, the connection and the associated socket are closed without a call to `SQLDisconnect`.

`TTC_ConnectTimeout` can be set in either the client connection string or the client DSN.

For more details, see "Choose SQL and PL/SQL timeout values" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Also see "TTC_Timeout" on page 2-138.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_ConnectTimeout` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_ConnectTimeout`	Seconds to wait for a client connect or disconnect request. Default is 20 seconds.
Windows ODBC Data Source Administrator	Not applicable	

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.

See "Using automatic client failover" in *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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_FailoverPortRange` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`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 on the Oracle TimesTen Client DSN Setup dialog.	Specify a lower value and an upper value for the port numbers in the format *lowervalue–uppervalue*.

TTC_NoReconnectOnFailover

Specifies whether the TimesTen client should *not* automatically reconnect to the server after a failover. If this is set to 1 (enabled), TimesTen is instructed to do all the usual client failover processing except for the reconnect. (For example, statement and connection handles are marked as invalid.) This is useful if the application does its own connection pooling or manages its own reconnection to the database after failover. The default value is 0 (reconnect).

You must configure automatic client failover for this option. See "Client connection failover in the *Oracle TimesTen In-Memory Database Scaleout User's Guide* for more information.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_NoReconnectOnFailover` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_NoReconnectOnFailover`	`0` = Client reconnects to server after failover (default). `1` = Client does all the failover processing, but does not reconnect after failover.
Windows ODBC Data Source Administrator	**No reconnect on failover** field on the Oracle TimesTen Client DSN Setup dialog.	`0` = Client reconnects to server after failover (default). `1` = Client does all the failover processing, but does not reconnect after failover.

TTC_Random_Selection

Specifies that the TimesTen client, if necessary, selects an alternative server from the list provided in `TTC_ServerN` attribute settings. If the client cannot connect to the selected server, it keeps redirecting until it successfully connects to one of the listed servers. If the client cannot connect to any of the selected servers, TimesTen returns an error.

- `1` (default): Initially, the list of failover servers provided by `TTC_ServerN` connection attributes is randomized. After which, the client selects sequentially from the randomized list for the initial connection and then for any client failover request.

- `0`: Client selects the first server specified by the `TTC_ServerN` connection attributes.

See "Using automatic client failover" in *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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_Random_Selection` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_Random_Selection`	`0` = Client selects the first server specified by the `TTC_ServerN` attributes. `1` (default)= Initially, the list of failover servers provided by `TTC_ServerN` connection attributes is randomized. After which, the client selects sequentially from the randomized list for the initial connection and then for any client failover request.
Windows ODBC Data Source Administrator	Not applicable	

TTC_REDIRECT

For TimesTen Scaleout, `TTC_REDIRECT` defines how a client is redirected. If this is set to 0 and the initial connection attempt to the desired data instance fails, then an error is returned and there are no further connection attempts. This does not affect subsequent failovers on that connection.

Automatic redirection: By default, this connection attribute is set to 1 so that a client connection is automatically redirected to any available data instance within the grid if the current host is busy or unavailable. The connection is redirected to the host with the fewest number of client connections.

Elements within a single replica set: If you want the client to connect to elements within a single replica set (because the data you are interested in is contained within this replica set), then set the `TTC_REDIRECT` connection attribute to 0. Then, the client connects only to the host indicated by the DSN or to the host with the element in the same replica set. If the connection is rejected, then a connection error is returned.

The `TTC_Redirect_Limit` attribute limits how many times the client is redirected. The number of hosts in your grid can be of a size that you want to limit the number of redirected client connection attempts for performance reasons. You can set the `TTC_Redirect_Limit` connection attribute to the number of connection redirection attempts. For example, setting `TTC_Redirect_Limit=10` limits the number of client connection redirection attempts to other hosts to 10 attempts. If the client does not connect within this number of attempts, a connection error is returned.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_REDIRECT` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_REDIRECT`	1 (default) - redirect to any available data instance 0 - error if redirection to specified data instance fails
Windows ODBC Data Source Administrator	Not applicable	

TTC_Redirect_Limit

For TimesTen Scaleout, `TTC_Redirect_Limit` limits how many times the client is redirected.

This is useful if the number of hosts in your grid is such that you want to limit the number of redirected client connection attempts for performance reasons.

For example, setting `TTC_Redirect_Limit=10` limits the number of client connection redirection attempts to other hosts to 10 attempts. If the client does not connect within this number of attempts, a connection error is returned.

> **Note:** There is no setting for no limit, but you can set it to a very large integer.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is not supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_Redirect_Limit` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_Redirect_Limit`	Integer to limit how many times the client is redirected. The default is 1.
Windows ODBC Data Source Administrator	Not applicable	.

TTC_RollbackRequiredOnFailover

For TimesTen Classic, indicates if the application must roll back if a client failover occurs. The default setting of 1 (enabled) results in a "transaction must roll back" error on the connection handle and all associated statement handles if failover occurs in the middle of a transaction. This explicitly indicates that the SQL operation may have failed. The application must roll back the transaction before proceeding.

The setting of 0 does not indicate any failures, which could also result in silent data loss. Only use the setting of 0 if necessary for backward compatibility.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set TTC_RollbackRequiredOnFailover as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	TTC_RollbackRequiredOnFailover	1 (default) - rollback required 0 - no rollback required
Windows ODBC Data Source Administrator	**Rollback required on failover** field on the Oracle TimesTen Client DSN Setup dialog.	1 (default) - rollback required 0 - no rollback required

TTC_Server or TTC_Server1

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. The network address provided can be a domain name server (DNS), host name or IP address.

> **Note:** If you are configuring for client failover, you may define more than one TimesTen Server. See "TTC_Server2, TTC_ServerN" on page 2-131 for more information.

If the TimesTen Server is listening on a non-default port number, you must provide the port number in one of the following ways:

- You can specify the port number within the `TTC_Server` connection attribute using:

  ```
  TTC_SERVER=server_host_name/server_port;
  ```

- You can specify the port number in the ODBC connection string.

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

  Or:

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

- If using TimesTen Classic, you can specify the port number within the logical server definition in the `ttconnect.ini` file.

See "TCP_Port" on page 2-120 for more details.

Note that:

- TimesTen Scaleout: You can use either `TTC_Server` or `TTC_Server1` for TimesTen Scaleout. If you define the `TTC_Server` connection attribute, the value is used only for the initial connection.

- TimesTen Classic: You can specify a logical server name for the `TTC_Server` attribute with TimesTen Classic that contains the network address and port number pair in the `ttconnect.ini` file. Once the logical server name is defined in the `ttconnect.ini` file, you can use that name as the value for the `TTC_Server` attribute in a Client DSN definition. 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. See "Creating and configuring a logical server name on Linux and UNIX" in the *Oracle TimesTen In-Memory Database Operations Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_Server` or `TTC_Server1` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_Server`, `TTC_Server1`	Character string specifying the logical server.
Windows ODBC Data Source Administrator	**Server Name or Network Address** field on the Oracle TimesTen Client DSN Setup dialog.	Character string specifying the logical server.

TTC_Server2, TTC_ServerN

For TimesTen Classic, this attribute specifies the logical server name to use if an automatic failover occurs. See the description of TTC_Server or TTC_Server1 for details on setting the value of this attribute and associated attributes.

- When using automatic client failover with an active standby pair replication scheme in TimesTen Classic, you can only define `TTC_Server2`. After which, the client alternately attempts to connect to `TTC_Server` and `TTC_Server2` until a connection succeeds or the `TTC_TIMEOUT` attribute expires.

- For other types of automatic client failover, you can specify a list of failover servers with `TTC_ServerN` connection attributes where `N >= 2`. TimesTen can iterate through this list of designated failover servers (as necessary) that you configured as `TTC_Server2`, `TTC_Server3`, `TTC_Server4`, and so on. The maximum number of servers that the client can specify is 999.

> **Note:** See "TTC_Random_Selection" on page 2-125 on how TimesTen iterates through the list of designated failover servers.

Unspecified values for `TTC_ServerN`, `TTC_Server_DSNN`, and `TCP_PortN` inherit the value of `TTC_Server` (or equivalently, `TTC_Server1`), `TTC_Server_DSN` (or `TTC_Server_DSN1`), and `TCP_PORT` (or `TCP_PORT1`), respectively. For example, if `TTC_Server2` is specified but `TTC_Server_DSN2` and `TCP_Port2` are not, then `TTC_Server_DSN2` is set to the `TTC_Server_DSN` value and `TCP_Port2` is set to the `TCP_Port` value.

You should configure your failover servers sequentially. If you do skip a number when configuring your failover servers, then TimesTen automatically creates the missing definition and assigns it to the server identified by `TTC_Server`. In this case, your client could fail over to the same server multiple times.

When using an active standby pair replication scheme for client failover, the TTC_Server or TTC_Server1 and `TTC_Server2` connection attributes could potentially have the same setting if it is a virtual IP address. Virtual IP addresses can dynamically move to different hosts; thus, both connection attributes may have the same definition, but could be referencing distinct databases.

See "Using automatic client failover" in *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.

Usage in TimesTen Scaleout and TimesTen Classic

These attributes are supported in TimesTen Classic.

These attributes are supported in TimesTen Scaleout.

Setting

Set `TTC_Server2` or `TTC_ServerN` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_Server2, TTC_ServerN`	Character string specifying the logical server to be used if an automatic failover occurs.
Windows ODBC Data Source Administrator	**Failover Server Name or Network Address** field on the Oracle TimesTen Client DSN Setup dialog configures the `TTC_Server2` connection attribute.	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.

More details on this topic can be found in "Creating Client DSNs on a TimesTen Client system" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_Server_DSN` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_Server_DSN`	Character string specifying the DSN that resides on the Server.
Windows ODBC Data Source Administrator	**Server DSN** field on the Oracle TimesTen Client DSN Setup dialog.	Character string specifying the DSN that resides on the Server.

TTC_Server_DSN2, TTC_Server_DSNn

For TimesTen Classic, this attribute specifies the Server DSN on the computer running the TimesTen Server. (This is unnecessary for TimesTen Scaleout.) 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` or `TTC_Server_DSNn`.

Unspecified values for `TTC_Server_DSNn` inherit the value of `TTC_Server_DSN` (or `TTC_Server_DSN1`). For example, if `TTC_Server2` is specified but `TTC_Server_DSN2` and `TCP_Port2` are not, then `TTC_Server_DSN2` is set to the `TTC_Server_DSN` value and `TCP_Port2` is set to the `TCP_Port` value.

See "Using automatic client failover" in *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.

Usage in TimesTen Scaleout and TimesTen Classic

These attributes are supported in TimesTen Classic.

These attributes are supported in TimesTen Scaleout.

Setting

Set `TTC_Server_DSN2` or `TTC_Server_DSNn` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or `Java programs or UNIX and Linux systems ODBC`.INI file	`TTC_Server_DSN2`, `TTC_Server_DSNn`	Character string specifying the DSN that resides on the Server to be used if an automatic failover occurs.
Windows ODBC Data Source Administrator	**Failover Server DSN** field (only for `TTC_Server_DSN2`) on the Oracle TimesTen Client DSN Setup dialog.	Character string specifying the DSN that resides on the Server to be used if an automatic failover occurs.

TTC_TCP_KEEPALIVE_INTVL_MS

The `TTC_TCP_KEEPALIVE_INTVL_MS` attribute sets the time interval (in milliseconds) between subsequential probes.

By default, if the connection fails, TimesTen Scaleout sends the client connection to another active server. Part of the method to see if the connection is up or if it has failed is to check the TCP socket. When a TCP connection is started, a set of timers are associated with the connection. These timers indicate when TimesTen Scaleout checks the TCP socket to determine whether the connection is up or if it has failed.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_TCP_KEEPALIVE_INTVL_MS` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_TCP_KEEPALIVE_INTVL_MS`	A positive integer value. Default is `1000`.
Windows ODBC Data Source Administrator	**TCP KeepAlive Interval** field on the Oracle TimesTen Client DSN Setup dialog.	A positive integer value. Default is `1000`.

TTC_TCP_KEEPALIVE_PROBES

The `TTC_TCP_KEEPALIVE_PROBES` attribute sets the number of unacknowledged probes to send before considering the connection as failed and notifying the client.

By default, if the connection fails, TimesTen Scaleout sends the client connection to another active server. Part of the method to see if the connection is up or if it has failed is to check the TCP socket. When a TCP connection is started, a set of timers are associated with the connection. These timers indicate when TimesTen Scaleout checks the TCP socket to determine whether the connection is up or if it has failed.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_TCP_KEEPALIVE_PROBES` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_TCP_KEEPALIVE_PROBES`	A positive integer value. Default is 2.
Windows ODBC Data Source Administrator	**TCP KeepAlive Probes** field on the Oracle TimesTen Client DSN Setup dialog.	A positive integer value. Default is 2.

TTC_TCP_KEEPALIVE_TIME_MS

The `TTC_TCP_KEEPALIVE_TIME_MS` attribute sets the duration time (in milliseconds) between the last data packet sent and the first probe.

By default, if the connection fails, TimesTen Scaleout sends the client connection to another active server. Part of the method to see if the connection is up or if it has failed is to check the TCP socket. When a TCP connection is started, a set of timers are associated with the connection. These timers indicate when TimesTen Scaleout checks the TCP socket to determine whether the connection is up or if it has failed.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `TTC_TCP_KEEPALIVE_TIME_MS` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`TTC_TCP_KEEPALIVE_TIME_MS`	A positive integer value. Default is `1000`.
Windows ODBC Data Source Administrator	**TCP KeepAlive Time** field on the Oracle TimesTen Client DSN Setup dialog.	A positive integer value. Default is `1000`.

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. For example, if the Client application is running long queries, you may want to increase the timeout interval.

The operating system select() call on the client side of a client/server connection uses the value of TTC_Timeout. The SQLExecute() and OCIStmtExecute() functions do not.

A value of 0 indicates that client/server operations should not timeout. 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 and socket closed (without a call to SQLDisconnect).

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

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 (the setting ignored) when shared memory is used for Client/Server inter-process communication.

See "Choose SQL and PL/SQL timeout values" in *Oracle TimesTen In-Memory Database Operations Guide* for information about the relationship between TTC_Timeout, SQLQueryTimeout, and PLSQL_TIMEOUT.

TTC_Timeout can be overridden for connect and disconnect requests by TTC_ConnectTimeout. See "TTC_ConnectTimeout" on page 2-122.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set TTC_Timeout as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	TTC_Timeout	A value between 0 and 99999 that represents the number of seconds that the TimesTen Client waits for an operation to complete before timing out. (The default value is 60.) In an active standby pair failover scenario, the minimum value is 60.

Where to set the attribute	How the attribute is represented	Setting
Windows ODBC Data Source Administrator	**Network Timeout** field on the Oracle TimesTen Client DSN Setup dialog.	A value between 0 and 99999 that represents the number of seconds that the TimesTen Client waits for an operation to complete before timing out. (The default value is 60.) In an active standby pair failover scenario, the minimum value is 60.

Wallet

The Wallet attribute specifies the fully qualified directory path name, where you placed the certificates that you generated (preferably the same directory path as on the client). There is no default location. If you specify a relative path, it is relative to the *timesten_home*/info directory.

You must set this attribute for both the client and the server. You must set the same path on both the client and server.

See "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set Wallet as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic or in the database definition (.dbdef) file in TimesTen Scaleout	Wallet	A fully qualified directory path name (no default).
Windows ODBC Data Source Administrator	**Wallet** field on the Oracle TimesTen Client DSN Setup dialog.	A fully qualified directory path name (no default).

TimesTen Server connection attributes

Server connection attributes are specified in the Server DSN only and are read at first connection. See "Defining Server DSNs for TimesTen Server on a Linux or UNIX 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 2–11, " TimesTen Server connection attributes" and described in detail in this section.

CipherSuites

The `CipherSuites` attribute lists the cipher suite or suites that can be used, depending also on the client setting. Specify `SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`, `SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`, or both, comma-separated and in order of preference. There is no default setting.

You must set this attribute for both the client and the server. For TLS to be used, the server and client settings must include at least one common suite.

See "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `CipherSuites` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	CipherSuites	Specify `SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`, `SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`, or both, comma-separated and in order of preference
Windows ODBC Data Source Administrator	**Encryption** field on the Oracle TimesTen Client DSN Setup dialog.	Specify `SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`, `SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`, or both, comma-separated and in order of preference

Encryption

The `Encryption` attribute specifies whether encryption is required or not for a client server connection.

You must set this attribute for both the client and the server. Cipher settings must be the same on both the client and server, in most cases.

See "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `Encryption` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`Encryption`	■ `accepted`: Enable an encrypted session if required or requested by the client; use an unencrypted session otherwise. (default) ■ `rejected`: Demand an unencrypted session. (If the server does not support encryption, TimesTen behaves as if this is the setting on the server.) The connection is rejected if the client requires encryption. ■ `requested`: Request an encrypted session if the client allows it (if the client has any setting other than `rejected`); use an unencrypted session otherwise. ■ `required`: Demand an encrypted session. Reject the connection if the client rejects encryption.
Windows ODBC Data Source Administrator	**Encryption** field on the Oracle TimesTen Client DSN Setup dialog.	■ `accepted`: Enable an encrypted session if required or requested by the client; use an unencrypted session otherwise. (default) ■ `rejected`: Demand an unencrypted session. (If the server does not support encryption, TimesTen behaves as if this is the setting on the server.) The connection is rejected if the client requires encryption. ■ `requested`: Request an encrypted session if the client allows it (if the client has any setting other than `rejected`); use an unencrypted session otherwise. ■ `required`: Demand an encrypted session. Reject the connection if the client rejects encryption.

SSLClientAuthentication

The `SSLClientAuthentication` attribute specifies whether SSL client authentication is required (setting of 1) or not (setting of 0, the default). With client authentication, the server validates an identity presented by the client, and requires an identity (public/private key) in the client wallet.

Regardless of the client authentication setting, server authentication is performed, where the client validates the server.

You must set this attribute for both the client and the server. Regardless of the client authentication setting, server authentication is performed, where the client validates the server.

See "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `SSLClientAuthentication` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`SSLClientAuthentication`	`1` - Client authentication is required `0` - No authentication is required. (the default)
Windows ODBC Data Source Administrator	**SSL Client Authentication** field on the Oracle TimesTen Client DSN Setup dialog.	`1` - Client authentication is required `0` - No authentication is required. (the default)

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`. For full details on `MaxConnsPerServer`, see "Connection attributes for Data Manager DSNs or Server DSNs" in the *Oracle TimesTen In-Memory Database Operations Guide*.

The value of this attribute takes precedence over the setting of the value of the `max_conns_per_server` attribute in the `timesten.conf` 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 6, "System Limits".

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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `MaxConnsPerServer` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	MaxConnsPerServer	A value between 1 and 2047. The default is `1`, which indicates that each connection has its own separate process, not just a separate thread within a process.
Windows ODBC Data Source Administrator	Not applicable	

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. See "Connection attributes for Data Manager DSNs or Server DSNs" in the *Oracle TimesTen In-Memory Database Operations Guide* for more details.

The value of this attribute takes precedence over the setting of the value of the `servers_per_dsn` attribute in the `timesten.conf` file. For details, see "Specifying multiple connections to the TimesTen Server" in the *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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `ServersPerDSN` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	`ServersPerDSN`	A value between 1 and 2047. The default is 1.
Windows ODBC Data Source Administrator	Not applicable	

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.

You generally should not need to set the `ServerStackSize` attribute. However, if the `ttcserver` process is getting repeatable Access Violations (Windows) or core dumps (Linux and Unix), you may consider increasing the `ServerStackSize` attribute to 1024 KB or greater.

This value of this attribute takes precedence over the setting of the `server_stack_size` attribute in the `timesten.conf` 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `ServerStackSize` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	ServerStackSize	Valid values depend on the platform. The default is 768 KB. If the `sysconf` call is available, the minimum is: `sysconf(_SC_THREAD_STACK_MIN)/ 1024` else the minimum is 0.
Windows ODBC Data Source Administrator	Not applicable	

SSLRenegotiationPeriod

The `SSLRenegotiationPeriod` attribute specifies a period of time, in minutes, after which session renegotiation is performed. The default setting is 0, meaning do not renegotiate based on a time period.

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

If both `SSLRenegotiationSize` and `SSLRenegotiationPeriod` are set with non-zero values, whichever setting occurs first causes the renegotiation.

For more details see "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set `SSLRenegotiationPeriod` as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic	`SSLRenegotiationPeriod`	An integer representing the number of minutes to wait to for session renegotiation. `0` - indicates that the time period is not used to renegotiate session start (the default)
Windows ODBC Data Source Administrator	Not applicable	

SSLRenegotiationSize

The SSLRenegotiationSize attribute specifies a number of megabytes of data transfer in either direction between the client and server, after which session renegotiation is performed. The default setting is 0, meaning do not renegotiate based on megabytes transferred.

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

If both SSLRenegotiationSize and SSLRenegotiationPeriod are set with non-zero values, whichever setting occurs first causes the renegotiation.

For more details see "Configuration for TLS for Client Server" in the *Oracle TimesTen In-Memory Database Security 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.

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is not supported in TimesTen Scaleout.

Setting

Set SSLRenegotiationSize as follows:

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems odbc.ini file in TimesTen Classic	SSLRenegotiationSize	An integer representing the number of megabytes of data transfer after which the session restart is renegotiated. 0 - indicates that the size is not used to renegotiate session start (the default)
Windows ODBC Data Source Administrator	Not applicable	

Wallet

The `Wallet` attribute specifies the fully qualified directory path name, where you placed the certificates that you generated (preferably the same directory path as on the client). There is no default location. If you specify a relative path, it is relative to the *timesten_home*/info directory.

You must set this attribute for both the client and the server. You must set the same path on both the client and server.

See "Configuration for TLS for Client/Server" in the *Oracle TimesTen In-Memory Database Security Guide* for more details.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This attribute is supported in TimesTen Classic.

This attribute is supported in TimesTen Scaleout.

Setting

Set `Wallet` as follows.

Where to set the attribute	How the attribute is represented	Setting
C or Java programs or UNIX and Linux systems `odbc.ini` file in TimesTen Classic or in the database definition (`.dbdef`) file in TimesTen Scaleout	Wallet	A fully qualified directory path name (no default).
Windows ODBC Data Source Administrator	**Wallet** field on the Oracle TimesTen Client DSN Setup dialog.	A fully qualified directory path name (no default).

3

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 `ttOptSetFlag` 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 "Dynamic SQL in PL/SQL (EXECUTE IMMEDIATE statement)" in the *Oracle TimesTen In-Memory Database PL/SQL Developer's Guide* for more details on this statement.

TimesTen built-in procedures that return result sets are not supported directly through OCI. You can use PL/SQL for this purpose. For an example, see "Use of PL/SQL in OCI to call a TimesTen built-in procedure" in the *Oracle TimesTen In-Memory Database C Developer's Guide*.

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;
        /

PL/SQL procedure successfully completed.
```

You can also call built-in procedures from the `ttIsql` command line:

```
Command> call ttDBCompactConfig(2000,5,2000);
< 2000, 5, 2000 >
1 row found.
```

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

List of built-in procedures

This section lists the built-in procedures available in TimesTen

Table 3–1   TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttAgingLRUConfig	Sets the Least Recently Used (LRU) aging attributes on all regular tables that have been defined with an LRU aging policy.	Yes	No
ttAgingScheduleNow	Starts the aging process	Yes	No
ttApplicationContext	Sets application-defined context for the next update record to pass application specific data to XLA readers.	Yes	No
ttBackupStatus	Returns information about the current or last backup of the database.	Yes	Yes
ttBlockInfo	Provides information about perm blocks and the amount of block-level fragmentation in a database.	Yes	Yes
ttBookmark	Returns information about the TimesTen transaction log.	Yes	Yes
ttCacheADGStandbyStateGet	Returns the state for the standby Oracle database in an Active Data Guard configuration.	Yes	No
ttCacheADGStandbyStateSet	Sets the state of the standby Oracle database in an Active Data Guard configuration.	Yes	No
ttCacheADGStandbyTimeoutGet	Retrieves the timeout value of the Oracle database in an Active Data Guard configuration.	Yes	No
ttCacheADGStandbyTimeoutSet	Sets the timeout value of the standby Oracle database in an Active Data Guard configuration.	Yes	No
ttCacheAllowFlushAwtSet	Enables you to execute a `FLUSH CACHE GROUP` statement on an AWT cache group.	Yes	No

Table 3–1   (Cont.) TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttCacheAutorefIntervalStatsGet	Returns statistical information about the last 10 autorefresh cycles for a specified autorefresh interval.	Yes	No
ttCacheAutorefresh	Starts an immediate autorefresh on a set of cache groups.	Yes	No
ttCacheAutorefreshLogDefrag	Compacts the trigger log space for a cache autorefresh table.	Yes	No
ttCacheAutorefreshStatsGet	Returns information about the last 10 autorefresh transactions on the specified cache group.	Yes	No
ttCacheAutorefreshSelectLimit	Configures the incremental autorefresh on a specific number of rows.	Yes	No
ttCacheAutorefreshXactLimit	Starts an immediate autorefresh on single table cache groups within a specified autorefresh interval and commits after the specified number of operations.	Yes	No
ttCacheAWTMonitorConfig	Sets AWT cache group monitoring.	Yes	No
ttCacheAWTThresholdGet	Returns the current transaction log file threshold for databases that include AWT cache groups.	Yes	No
ttCacheAWTThresholdSet	Sets the threshold for the number of transaction log files that can accumulate before AWT is considered terminated or too far behind to catch up.	Yes	No
ttCacheCheck	Checks for missing constraints for cached tables on the Oracle database	Yes	No
ttCacheConfig	Configures timeout value and recovery policies for cache groups.	Yes	No
ttCacheConnPoolApply	Applies the cache connection pool settings.	Yes	No
ttCacheConnPoolGet	Retrieves the cache connection pool settings.	Yes	No
ttCacheConnPoolSet	Configures the cache connection pool for dynamic cache groups.	Yes	No
ttCacheDbCgStatus	Returns the automatic refresh status of the database and the specified cache group.	Yes	No
ttCacheDDLTrackingConfig	Configures tracking of DDL statements issued on cached Oracle database tables.	Yes	No

Table 3–1 (Cont.) TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttCachePolicyGet	Returns the current policy used to determine when the TimesTen cache agent for the connected database should run.	Yes	No
ttCachePolicySet	Sets the policy used to determine when the TimesTen cache agent for the connected database should run.	Yes	No
ttCachePropagateFlagSet	Configures propagation of committed updates to a cache group within the current transaction to the Oracle database.	Yes	No
ttCacheSqlGet	Generates the Oracle SQL statements to install or uninstall Oracle database objects for certain types of cache groups.	Yes	No
ttCacheStart	Starts the TimesTen cache agent.	Yes	No
ttCacheStop	Stops the TimesTen cache agent.	Yes	No
ttCacheUidGet	Returns the cache administration user ID.	Yes	No
ttCacheUidPwdSet	Sets the cache administration user ID and password.	Yes	No
ttCkpt	Performs a non-blocking checkpoint operation.	Yes	No
ttCkptBlocking	Performs a blocking checkpoint operation.	Yes	No
ttCkptConfig	Reconfigures the background checkpointer dynamically or returns the currently active settings of the configuration parameters.	Yes	No
ttCkptHistory	Returns information about the last eight checkpoints.	Yes	No
ttCommitBufferStats	Returns the number of commit buffer overflows and the high watermark for memory used by transaction reclaim records during transaction commit process.	Yes	Yes
ttCommitBufferStatsReset	Resets transaction commit buffer statistics to 0.	Yes	No
ttCompact	Compacts both the permanent and temporary data partitions of the database.	Yes	No
ttComputeTabSizes	Refreshes table size statistics stored in TimesTen system tables.	Yes	Yes

Table 3–1   (Cont.) TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttConfiguration	Returns the values for most, but not all, connection attributes for the current database connection.	Yes	Yes
ttContext	Returns the context value of the current connection.	Yes	Yes
ttDataStoreStatus	Returns the list of processes connected to a database.	Yes	Yes
ttDBCompactConfig	Sets or returns the value of a TimesTen database system parameter.	Yes	Yes
ttDBConfig	Sets or returns the value of a TimesTen database system parameter.	Yes	Yes
ttDBWriteConcurrencyModeGet	Returns information about the write concurrency mode of the database and the status of write concurrency mode operations and transitions.	Yes	No
ttDBWriteConcurrencyModeSet	Controls read optimization during periods of concurrent write operations.	Yes	No
ttDistributionProgress	Returns a progress report of an ongoing redistribution process.	No	Yes
ttDurableCommit	Sets transaction durability.	Yes	Yes
ttEpochCreate	Causes the next committed transaction in a grid to commit as an epoch transaction.	No	Yes
ttEpochSessionGet	Returns the epoch identifier of the last epoch created by the current connection.	No	Yes
ttHeapInfo	Returns information about the size and usage of heap memory.	Yes	Yes
ttHostNameGet	Returns the name of the current local host.	Yes	No
ttHostNameSet	Specifies the name of the default local host	Yes	No
ttIndexAdviceCaptureDrop	Drops existing capture data for either the current connection	Yes	Yes
ttIndexAdviceCaptureEnd	Ends either an active connection level capture from the current connection or an active database level capture	Yes	Yes
ttIndexAdviceCaptureInfoGet	Returns information for each active capture.	Yes	Yes
ttIndexAdviceCaptureOutput	Returns index recommendations from the last recorded capture at the specified level.	Yes	Yes
ttIndexAdviceCaptureStart	Enables index advice capture.	Yes	Yes

Table 3–1 (Cont.) TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttLatchStatsGet	Displays latch statistics.	Yes	Yes
ttLoadFromOracle	Executes a query on the Oracle database and loads the result into a TimesTen table.	Yes	Yes
ttLockLevel	Changes the lock level between row-level and database-level locking on the next transaction and for all subsequent transactions for the connection.	Yes	Yes
ttLockWait	Changes the lock timeout interval of the current connection.	Yes	Yes
ttLogHolds	Returns information about transaction log holds	Yes	Yes
ttMonitorHighWaterReset	Changes the value of the `PERM_IN_USE_HIGH_WATER` column in the `MONITOR` system table to the value of the `PERM_IN_USE_SIZE` and sets the value of the `TEMP_IN_USE_HIGH_WATER` column to the current value of `TEMP_IN_USE_SIZE` column.	Yes	Yes
ttOptClearStats	Clears the statistics for the specified table.	Yes	Yes
ttOptCmdCacheInvalidate	Forces a recompilation should a dependent command be invoked again, or removes the command from the cache. It must be re-prepared by the user.	Yes	Yes
ttOptEstimateStats	Updates the statistics for the specified table.	Yes	Yes
ttOptGetColStats	Returns statistics information in text format.	Yes	Yes
ttOptGetFlag	Returns the optimizer flag settings for the current transaction.	Yes	Yes
ttOptGetMaxCmdFreeListCnt	Returns the size of the free list of SQL compiled command cache.	Yes	Yes
ttOptGetOrder	Returns a single-row result set containing the join order for the current transaction.	Yes	Yes
ttOptSetColIntvlStats	Modifies the statistics for the specified columns with interval information.	Yes	Yes
ttOptSetColStats	Modifies the statistics for the specified columns.	Yes	Yes
ttOptSetFlag	Sets flags to alter the generation of execution plans by the TimesTen query optimizer.	Yes	Yes

Table 3–1   (Cont.) TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttOptSetMaxCmdFreeListCnt	Sets the maximum count of the free list of SQL compiled commands for regular tables.	Yes	Yes
ttOptSetMaxPriCmdFreeListCnt	Sets the maximum count of the free list of SQL compiled commands that perform materialized view maintenance.	Yes	Yes
ttOptSetOrder	Specifies the order in which tables should be joined by the optimizer.	Yes	Yes
ttOptSetTblStats	Modifies the statistics for the specified table.	Yes	Yes
ttOptShowJoinOrder	Returns the join order of the last prepared or executed SQL statement in the current transaction.	Yes	Yes
ttOptStatsExport	Returns the set of statements required to restore the table statistics to the current state.	Yes	Yes
ttOptUpdateStats	Updates the statistics for the specified table.	Yes	Yes
ttOptUseIndex	Alters the generation of execution plans by the TimesTen query optimizer.	Yes	Yes
ttPLSQLMemoryStats	Returns result statistics about PL/SQL library cache performance and activity.	Yes	Yes
ttRamPolicyAutoReloadGet	Returns the RAM autoreload policy used to determine if a database is reloaded into RAM after an invalidation.	Yes	No
ttRamPolicyAutoReloadSet	Determines the RAM autoreload policy if a database is invalidated.	Yes	No
ttRamPolicyGet	Returns the RAM policy used to determine when a database is loaded into memory.	Yes	No
ttRamPolicySet	Defines the policy used to determine when a database is loaded into memory.	Yes	No
ttRedundantIndexCheck	Scans tables to find redundant indexes.	Yes	Yes
ttRepDeactivate	Changes the state of the active database in an active standby pair from `ACTIVE` to `IDLE`.	Yes	No
ttReplicationStatus	Returns the status of one or more replication peer databases.	Yes	No
ttRepPolicyGet	Returns the replication restart policy	Yes	No
ttRepPolicySet	Specifies the replication restart policy	Yes	No

Table 3–1   (Cont.) TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttRepQueryThresholdGet	Returns the number of seconds that was most recently specified as the query threshold for the replication agent.	Yes	No
ttRepQueryThresholdSet	specifies the number of seconds that a query can be executed by the replication agent before TimesTen writes a warning to the daemon log.	Yes	No
ttRepStart	Starts the TimesTen replication agent for the connected database.	Yes	No
ttRepStateGet	Returns the current replication state of a database in an active standby pair.	Yes	No
ttRepStateSave	Saves the state of a remote peer database in an active standby pair to the currently connected database.	Yes	No
ttRepStateSet	Sets the replication state of a database in an active standby pair replication scheme.	Yes	No
ttRepStop	Stops the TimesTen replication agent for the connected database.	Yes	No
ttRepSubscriberStateSet	Changes a replicating subscriber's state with respect to the executing master store.	Yes	No
ttRepSubscriberWait	Causes the caller to wait until all transactions that committed before the call have been transmitted to the subscriber.	Yes	No
ttRepSyncGet	Returns static attributes associated with the caller's use of the replication- based return service.	Yes	No
ttRepSyncSet	Sets static attributes associated with the caller's use of the replication-based return service.	Yes	No
ttRepSyncSubscriberStatus	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.	Yes	No
ttRepTransmitGet	Returns the status of transmission of updates to subscribers for the current transaction.	Yes	No
ttRepTransmitSet	Updates on the connection it is executed in from being replicated to any subscriber.	Yes	No

Table 3–1   (Cont.) TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttRepXactStatus	Checks the status of a `RETURN RECEIPT` or `RETURN TWOSAFE` replication transaction.	Yes	No
ttRepXactTokenGet	Returns a token for `RETURN RECEIPT` or `RETURN TWOSAFE` replication transactions.	Yes	No
ttSetUserColumnID	Sets the value for the user-specified column ID.	Yes	No
ttSetUserTableID	Sets the value of the user table ID.	Yes	No
ttSize	Estimates the size of a table or view and the size of indexes.	Yes	Yes
ttSQLCmdCacheInfo	Returns information about all prepared SQL statements in the TimesTen SQL command cache.	Yes	Yes
ttSQLCmdCacheInfoGet	Returns information about the commands in the TimesTen SQL command cache.	Yes	Yes
ttSQLCmdQueryPlan	Returns all detailed runtime query plans for SQL statements in the TimesTen SQL command cache.	Yes	Yes
ttSQLExecutionTimeHistogram	Returns a histogram of SQL execution times.	Yes	Yes
ttStatsConfig	Controls statistics collection and parameters for the `ttStats` utility.	Yes	Yes
ttStatsConfigGet	Returns parameters of the ttStats utility that you can set with the `ttStatsConfig` built-in procedure.	Yes	Yes
ttTableSchemaFromOraQueryGet	Evaluates a `SELECT` query on a table in an Oracle database and generates a `CREATE TABLE SQL` statement that you can choose to execute.	Yes	Yes
ttVersion	Returns TimesTen release information.	Yes	Yes
ttWarnOnLowMemory	Specifies that operations executed on the current connection should return a warning if they allocate memory and find that memory is low.	Yes	Yes
ttXactIdGet	Returns transaction ID information for interpreting lock messages.	Yes	Yes
ttXlaBookmarkCreate	Creates the specified bookmark.	Yes	No
ttXlaBookmarkDelete	Deletes the specified bookmark.	Yes	No
ttXlaSubscribe	Configures persistent XLA tracking of a table.	Yes	No

Table 3–1   (Cont.) TimesTen built-in procedures

Built-in Name	Description	TimesTen Classic Support	TimesTen Scaleout Support
ttXlaUnsubscribe	Stops persistent XLA tracking of a table.	Yes	No

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. LRU aging enables you to maintain the amount of memory used in a TimesTen database within a specified threshold by deleting the least recently used data. Data is removed if the database space in-use exceeds the specified threshold values.

For cache groups, LRU aging is defined at the root table for the entire cache instance. LRU aging can be defined for all cache group types except for explicitly loaded autorefresh cache groups. LRU aging is defined by default on dynamic cache groups. For explicitly loaded cache groups, use time-based aging.

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.

Required privilege

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

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

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

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views:

SYS.GV$BACKUP_STATUS

SYS.V$BACKUP_STATUS

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.

Column	Type	Description
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.
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, 2018-01-12 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

This procedure returns a row for the element from which it was called. To see information about other elements, query the SYS.GV$BLOCK_INFO system table.

Related views

This procedure has these related views.

SYS.GV$BLOCK_INFO

SYS.V$BLOCK_INFO

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();
< 1537, 16, 236036720, 235991352 >
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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$BOOKMARK

SYS.V$BOOKMARK

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.

Column	Type	Description
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();
<379, 60193048,  379, 60192768, -1, -1>
1 row found.
```

ttCacheADGStandbyStateGet

Returns the state for the standby Oracle database that was specified with the
`ttCacheADGStandbyStateSet` built-in procedure.

Required privilege

This procedure requires no privileges.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttCacheADGStandbyStateGet()
```

Parameters

`ttCacheADGStandbyStateGet` has no parameters.

Result set

`ttCacheADGStandbyStateGet` returns the state of the standby Oracle database.

Parameter	Type	Description
ADGStandbyState	TT_VARCHAR(20)	`OK`: The standby Oracle database in an Active Data Guard configuration is considered to be up.
		`FAILED`: The standby Oracle database in the Active Data Guard configuration has failed. The cache agent does not try to contact the standby Oracle database and continues the autorefresh with only the primary Oracle database.

Examples

The following example shows how to call the `ttCacheADGStandbyStateSet` built-in
procedure to set the state of the standby Oracle database in an Active Data Guard
environment first to `OK` and then to `FAILED`. The `ttCacheADGStandbyStateGet` built-in
procedure retrieves the value of the current state of the standby Oracle database.

```
Command> call ttCacheADGStandbyStateSet('OK');
Command> call ttCacheADGStandbyStateGet();
< OK >
1 row found.

Command> call ttCacheADGStandbyStateSet('FAILED');
Command> call ttCacheADGStandbyStateGet();
< FAILED >
1 row found.
```

See also

      ttCacheADGStandbyStateSet
      ttCacheADGStandbyTimeoutGet
      ttCacheADGStandbyTimeoutSet

ttCacheADGStandbyStateSet

For an Active Data Guard environment, the user can call the
ttCacheADGStandbyStateSet built-in procedure to inform the cache agent of the state
of the standby Oracle database.

- Set the state of the standby Oracle database to OK and the cache agent
  autorefreshes only those transactions that have been replicated from the primary
  Oracle database to the standby Oracle database.

  - If the standby Oracle database fails and if you have set a timeout with the
    ttCacheADGStandbyTimeoutSet built-in procedure, then the state changes to
    FAILED if the standby Oracle database does not respond after the timeout is
    reached.

  - If the standby Oracle database fails and you did not set the timeout, then
    autorefresh stalls until the standby Oracle database recovers (unless you set
    the state of the standby Oracle database to FAILED).

- Set the state of the standby Oracle database to FAILED if you know the standby
  Oracle database has failed and it should not be used as part of the autorefresh.
  When you set the state to FAILED, the cache agent does not wait for transactions to
  be replicated to the standby Oracle database and continues the autorefresh with
  only the primary Oracle database. The cache agent does not include the standby
  Oracle database in the autorefresh, even if it has recovered and is currently active,
  until you change the state to OK.

Required privilege

This procedure requires no privileges.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttCacheADGStandbyStateSet('[OK | FAILED]')
```

Parameters

ttCacheADGStandbyStateSet has the parameters:

Parameter	Type	Description
ADGStandbyState	TT_VARCHAR(20)	OK: Tells the cache agent that the standby Oracle database in an Active Data Guard configuration is active. This is the default.
		FAILED: Tells the cache agent that the standby Oracle database in the Active Data Guard configuration has failed. The cache agent does not try to contact the standby Oracle database and continues the autorefresh with only the primary Oracle database.

Result set

ttCacheADGStandbyStateSet returns no results.

Examples

The following example shows how to call the ttCacheADGStandbyStateSet built-in procedure to set the state of the standby Oracle database in an Active Data Guard environment first to OK and then to FAILED. The ttCacheADGStandbyStateGet built-in procedure retrieves the value of the current state of the standby Oracle database.

```
Command> call ttCacheADGStandbyStateSet('OK');
Command> call ttCacheADGStandbyStateGet();
< OK >
1 row found.

Command> call ttCacheADGStandbyStateSet('FAILED');
Command> call ttCacheADGStandbyStateGet();
< FAILED >
1 row found.
```

See also

ttCacheADGStandbyStateGet
ttCacheADGStandbyTimeoutGet
ttCacheADGStandbyTimeoutSet

ttCacheADGStandbyTimeoutGet

Retrieve the timeout specified with the `ttCacheADGStandbyTimeoutSet` built-in procedure.

Required privilege

This procedure requires no privileges.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttCacheADGStandbyTimeoutGet()
```

Parameters

`ttCacheADGStandbyTimeoutGet` has no parameters.

Result Set

`ttCacheADGStandbyTimeoutGet` returns the *timeout*:

Parameter	Type	Description
ADGStandbyTimeout	TT_INTEGER	The timeout in seconds on how long to wait for a response from the standby Oracle database in an Active Data Guard configuration before using only the primary Oracle database to autorefresh the cache groups.

Examples

The following example shows how to use the `ttCacheADGStandbyTimeoutSet` built-in procedure to set the timeout that indicates the time to wait for a response from the standby Oracle database. And then, it shows how to call the `ttCacheADGStandbyTimeoutGet` built-in procedure to retrieve the value of the timeout.

```
Command> call ttCacheADGStandbyTimeoutSet('60');
Command> call ttCacheADGStandbyTimeoutGet();
< 60 >
1 row found.
```

See also

ttCacheADGStandbyStateSet
ttCacheADGStandbyStateGet
ttCacheADGStandbyTimeoutSet

ttCacheADGStandbyTimeoutSet

You can set a timeout with the ttCacheADGStandbyTimeoutSet built-in procedure to designate how long to wait for a response from the standby Oracle database in an Active Data Guard configuration. If the standby Oracle database does not respond after this period, then the state of the standby Oracle database is automatically changed to FAILED and the cache agent facilitates autorefresh using only the primary Oracle database.

> **Note:** At any time, the user can restore the standby Active Data Guard state by executing the ttCacheADGStandbyStateSet built-in procedure and set the state to OK.

Required privilege

This procedure requires no privileges.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttCacheADGStandbyTimeoutSet(timeout)
```

Parameters

ttCacheADGStandbyTimeoutSet has the parameters:

Parameter	Type	Description
ADGStandbyTimeout	TT_INTEGER NOT NULL	A timeout specified in seconds on how long to wait for the standby Oracle database to respond before using only the primary Oracle database to autorefresh the cache groups.
		Default is 0, which indicates that no timeout is used and the state of the standby Oracle database does not change from OK to FAILED as a result of this timeout.

Result set

ttCacheADGStandbyTimeoutSet returns no results.

Examples

The following example shows how to use the ttCacheADGStandbyTimeoutSet built-in procedure to set the timeout that indicates the time to wait for a response from the standby Oracle database. And then, it shows how to call the ttCacheADGStandbyTimeoutGet built-in procedure to retrieve the value of the timeout.

```
Command> call ttCacheADGStandbyTimeoutSet('60');
Command> call ttCacheADGStandbyTimeoutGet();
< 60 >
1 row found.
```

See also

ttCacheADGStandbyStateSet
ttCacheADGStandbyTimeoutGet
ttCacheADGStandbyStateGet

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttCacheAllowFlushAwtSet (enableFlush)
```

Parameters

`ttCacheAllowFlushAwtSet` has the parameters:

Parameter	Type	Description
allow	TT_INTEGER NOT NULL	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;
         CALL ttCacheAllowFlushAwtSet(1);
         FLUSH CACHE GROUP marketbasket;
         CALL ttCacheAllowFlushAwtSet(0);
         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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Column	Type	Description
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.
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 s 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
"Executing large transactions with incremental autorefresh 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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('CompactOnly');
```

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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

Column name	Column type	Description	Returned for full autorefresh
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
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.	N
		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.	
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.	N
		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.	
totalNumOracleBytes	TT_BIGINT	The total number of bytes transferred from the Oracle database since the cache agent started.	N

Column name	Column type	Description	Returned for full autorefresh
totalNumRootTblRows	TT_BIGINT	The total number of root table rows autorefreshed since the cache agent started.	Y
totalDuration	TT_BIGINT	The total autorefresh duration in milliseconds since the cache agent started.	Y
status	VARCHAR2 (128)	A string description of the status of the current autorefresh. See "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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

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

Parameters

ttCacheAutorefreshSelectLimit has the parameters:

Parameter	Type	Description
param	TT_VARCHAR2 (50) 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_VARCHAR2 (200)	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
param	TT_VARCHAR2(50)	The *autorefreshInterval* that designates the cache group (the one with this autorefresh interval value).
value	TT_VARCHAR2(200)	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 starts 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

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

Parameters

`ttCacheAutorefreshXactLimit` has the parameters:

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

"Executing large transactions with incremental autorefresh 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.
AWTSamplingFactor	TT_INTEGER NOT NULL	Positive integer that specifies the frequency with which the AWT workflow is sampled.

Examples

Example 3–1

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

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

Example 3–2

Enable monitoring and set the sampling frequency to 16.

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

Example 3–3

Disable monitoring.

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

See also

"ttRepAdmin" on page 5-111

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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
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 5-4

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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. By default, value is 40. ■ AutorefreshLogMonitorInterval - Specifies the time interval (in seconds) for how often to perform the percentage calculation of the cache administration user's tablespace usage. Default value is 300 seconds. ■ 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. ■ DisableFullAutorefresh - Disables or enables full autorefresh behavior. Default is 0, meaning that full autorefresh is enabled. The value of 1 indicates that full autorefresh is disabled. ■ 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.
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.

Parameter	Type	Description
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 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 AutoRefreshLogFragmentationWarningPCT, the value of the fourth parameter must be an integer between 1 and 100, representing a percentage of the table. The default is 40.
		■ When *Param* is AutorefreshLogMonitorInterval, it specifies the interval in seconds when to calculate the percentage of usage of the cache administration user's tablespace. The default is every 300 seconds.
		■ 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 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
		AutoRefreshLogFragmentationWarningPCT
		AutorefreshLogMonitorInterval
		AutoRefreshLogDeFragmentAction
		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*.

For the *Value* column:

- 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 AutoRefreshLogFragmentationWarningPCT, the value can be 0 to 100.
- When *Param* is AutorefreshLogMonitorInterval, the value can be an integer.
- When *Param* is AutoRefreshLogDeFragmentAction, the value can be Manual, CompactOnly or CompactandReclaim.
- 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');
```

TimesTen Classic calculates the percentage of fragmentation for the change log tables as a ratio of used space to the total size of the space. If this ratio falls below a defined threshold, TimesTen alerts you of the necessity for defragmentation of the change log tables by logging a message. By default, this threshold is set to 40%. For example, to set the fragmentation threshold to 50%, perform:

```
Command> CALL ttCacheConfig('AutoRefreshLogFragmentationWarningPCT',,,'50');
< AutoRefreshLogFragmentationWarningPCT, <NULL>, <NULL>, 50 >
1 row found.
```

To set the time interval to 3600 seconds for when to calculate the fragmentation percentage of the change log tables, perform:

```
Command> CALL ttCacheConfig('AutorefreshLogMonitorInterval',,,'3600');
< AutorefreshLogMonitorInterval, <NULL>, <NULL>, 3600 >
1 row found.
```

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 an active standby pair.

See also

ttCacheDbCgStatus
ttCachePolicyGet
ttCachePolicySet
ttCacheStart

ttCacheStop
ttCacheUidGet
ttCacheUidPwdSet
"ttAdmin" on page 5-4
"Managing a Caching Environment", "Reporting Oracle Database permanent errors for AWT cache groups", and "Setting up TimesTen Cache in an Oracle RAC environment" (regarding Agent Failover) in *Oracle TimesTen Application-Tier Database Cache User's Guide*.

ttCacheConnPoolApply

Description

This procedure enables you to dynamically resize the cache connection pool parameters on each child server process. After which, the cache connection pool parameters are associated with the child server process.

Use the `ChildServer` connection attribute to identify each child server process, where `ChildServer=`*n* and *n* is a number ranging from 1 to the number of running child server processes. Once connected to the child server process, you can execute the `ttCacheConnPoolApply` built-in procedure that is meant for a specific child server process.

For more details, see "Managing a cache connection pool to the Oracle database for dynamic load requests" in *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Required privilege

This procedure requires TimesTen cache administration manager or `Admin` privileges to execute.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttCacheConnPoolApply( )
```

Parameters

There are no parameters for this procedure.

Result Set

`ttCacheConnPoolApply` returns no results..

Examples

The following connects to the child server process identified as 1 and applies the saved cache connection pool configuration to this child server process. It does the same process for child server process 2 (given that `ServersPerDSN=2`).

```
Command> connect "DSN=cache1;ChildServer=1;";
Command> call ttCacheConnPoolApply;
Command> disconnect;

Command> connect "DSN=cache1;ChildServer=2;";
Command> call ttCacheConnPoolApply;
Command> disconnect;
```

Notes

This procedure is available only for TimesTen Cache.

You can only execute the `ttCacheConnPoolApply` built-in procedure from a multithreaded client/server connection.

If the cache connection pool fails, you can recreate the pool by executing the `ttCacheConnPoolApply` built-in procedure from any child server process.

See also

ttCacheConnPoolGet
ttCacheConnPoolSet

ttCacheConnPoolGet

Description

This procedure retrieves the current values of the cache connection pool parameters.

The `ChildServer` connection attribute identifies each child server process, where `ChildServer=n` and *n* is a number ranging from 1 to the number of running child server processes. Once connected to the child server process, you can execute the `ttCacheConnPoolGet('current')` built-in procedure that is meant for a specific child server process.

For more details, see "Managing a cache connection pool to the Oracle database for dynamic load requests" in *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Required privilege

This procedure requires TimesTen cache administration manager or `Admin` privileges to execute.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttCacheConnPoolGet(state)
```

Parameters

The `state` parameter can be either:

- `saved`: Returns the cache connection parameters that are saved on the Oracle database. When querying the saved configuration, you can execute the `ttCacheConnPoolGet` built-in procedure from a direct connection, a single-threaded client/server connection or a multithreaded client/server connection.

- `current`: Returns the cache connection parameters that have defined the cache connection pool for this current child server process. When querying the current configuration of a specific child server process, you can execute the `ttCacheConnPoolGet` built-in procedure only from a multithreaded client/server connection.

Result set

`ttCacheConnPoolGet` returns the following values:

Table 3–2   ttCacheConnPoolGet result set

Parameter	Description
MinSize	The set value for the minimum number of open connections for the cache connection pool.
MaxSize	The set value for the maximum number of open connections for the cache connection pool.
IncrSize	The set value for the increment by which the number of open connections increases when needed.
Timeout	A timeout (in seconds) for the connections in the cache connection pool. Connections that idle for more than this timeout are terminated to maintain an optimum number of open connections and returned to the cache connection pool.
ConnNoWait	A directive given whether to wait for an available connection from the cache connection pool when no connection is immediately available. Valid values are: ■ Disabled (0): Any dynamic load operations wait for an available connection in the cache connection pool before proceeding. ■ Enabled (1): Any dynamic load operations fail with an error if there is no available connection in the cache connection pool.
OpenCount	The current number of open connections in the cache connection pool. If you query the saved parameters, a -1 is displayed for this value.
BusyCount	The current number of busy connections in the cache connection pool. If you query the saved parameters, a value of -1 is displayed for this value.
LastOraErr	Displays the number of the last Oracle Database error returned (if applicable) when attempting to retrieve a connection from the cache connection pool. For example, a value of 1034 would indicate that "ORA-0134: ORACLE not available" error was the last Oracle Database error returned. When requesting current and no Oracle Database error is returned, a 0 is returned. If you query the saved parameters, a -1 is displayed for this value.

Examples

Query the values for the cache connection pool that are saved on the Oracle database.

```
Command> call ttCacheConnPoolGet('saved');
< 1, 10, 1, 10, 0, -1, -1, -1>
```

Notes

This procedure is available only for TimesTen Cache.

See also

ttCacheConnPoolApply
ttCacheConnPoolSet

ttCacheConnPoolSet

Description

This procedure sizes the cache connection pool to avoid contention for connections. This procedure saves the values of these parameters on the Oracle database, which are then used as the default values when restarting the TimesTen server.

If you are dynamically changing the sizing, you can apply the changes to each TimesTen server by executing the `ttCacheConnPoolApply` built-in procedure.

For more details, see "Managing a cache connection pool to the Oracle database for dynamic load requests" in *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Required privilege

This procedure requires TimesTen cache administration manager or `Admin` privileges to execute.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttCacheConnPoolSet(MinSize, MaxSize, IncrSize, Timeout, ConnNoWait)
```

Parameters

All parameters must be specified for the `ttCacheConnPoolSet` built-in procedure. You can execute the `ttCacheConnPoolSet` built-in procedure from a direct connection, a single-threaded client/server connection or a multithreaded client/server connection.

Table 3–3   ttCacheConnPoolSet input parameters

Parameter	Description
MinSize	The minimum number of open connections for the cache connection pool. The default is 10 connections. The minimum is 1; the maximum is 2000 connections.
MaxSize	The maximum number of open connections for the cache connection pool. The default is 32 connections. The minimum is 1; the maximum is 2000 connections.
IncrSize	The increment by which the number of open connections increases when needed. The default increment is 1. The minimum is 1; the maximum is 2000 connections.
Timeout	A timeout (in seconds) for the connections in the cache connection pool. Connections that idle for more than this timeout are terminated to maintain an optimum number of open connections and returned to the cache connection pool. The default is 100 seconds. If set to 0, then the connections never time out. The minimum is 0; the maximum is 300.

Table 3–3   (Cont.) ttCacheConnPoolSet input parameters

Parameter	Description
ConnNoWait	A directive given whether to wait for an available connection from the cache connection pool when no connection is immediately available. Valid values are:
	■ Disabled (0 - the default): Any dynamic load operations wait for an available connection in the cache connection pool before proceeding.
	■ Enabled (1): Any dynamic load operations fail with an error if there is no available connection in the cache connection pool.

Examples

The following initiates the minimum and maximum number of pooled connections to be between 10 and 32 connections and the increment is 1. The maximum idle time by the client is set to 10 seconds. And all dynamic load operations will wait for an available connection from the cache connection pool.

```
Command> call ttCacheConnPoolSet(10, 32, 1, 10, 0);
```

Notes

This procedure is available only for TimesTen Cache.

See also

ttCacheConnPoolGet
ttCacheConnPoolApply

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Column	Type	Value
cgStatus	VARCHAR2(20)	Specifies the autorefresh status of the specified cache group. The status is one of:
		ok - The specified cache group is synchronized with the Oracle database. The cache agent has been in contact with the Oracle 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 5-4

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 5-4

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Always specify the TimesTen database location as a full path. If a relative path is specified, TimesTen would look relative to the working directory of the daemon, *timesten_home*/info.

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 5-4

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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;
        call ttCachePropagateFlagSet(0);
        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;
        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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.
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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

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](#)

Start

CacheStart

ttCacheDbCgStatus
ttCachePolicyGet
ttCachePolicySet
ttCacheStop
ttCacheUidPwdSet
ttCacheUidGet
"ttAdmin" on page 5-4

-In Procedures   **3-79**

ttCacheStop

Description

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

Required privilege

This procedure requires the CACHE_MANAGER privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 5-4

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 5-4

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 an unencrypted 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 5-4

ttCkpt

Description

This procedure performs a nonblocking checkpoint operation. For information on blocking checkpoints, see "ttCkptBlocking" on page 3-87. 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Parameter	Type	Description
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.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure returns a row for the element from which it was called.

Related views

This procedure has these related views.

`SYS.GV$CKPT_CONFIG`

`SYS.V$CKPT_CONFIG`

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.

Parameter	Type	Description
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.

Also see "Displaying checkpoint history and status" in *Oracle TimesTen In-Memory Database Operations Guide*.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout this procedure returns a row for the element from which it was called.

Related views

This procedure has these related views.

SYS.GV$CKPT_HISTORY

SYS.V$CKPT_HISTORY

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.

Column	Type	Description
status	TT_CHAR (16) NOT NULL	Result status of the checkpoint operation. Value is one of:
		In `Progress` - The checkpoint is currently in progress. Only the most recent result row can have this status.
		`Completed` - The checkpoint completed successfully.
		`Failed` - The checkpoint failed. Only the most recent result row can have this status. In this case the error column indicates the reason for the failure.
initiator	TT_CHAR (16) NOT NULL	The source of the checkpoint request. Value is one of:
		`User` - A user-level application. This includes TimesTen utilities such as [ttIsql].
		`Checkpointer` - The background checkpointer.
		`Subdaemon` - The managing subdaemon of the database. For a shared database, the final disconnect checkpoint is taken by the subdaemon.
reason	TT_CHAR (16) NOT NULL	The reason for this checkpoint. For example: after database creation, after recovery, final checkpoint after shutdown, after the user runs a built-in procedure, or after a flush operation.
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.

Column	Type	Description
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.
ckptVNo	TT_INTEGER NOT NULL	The checkpoint sequence number that is incremented for each checkpoint.
logsPurged	TT_BIGINT	The number of log files purged by this checkpoint.
bookmarkName	TT_VARCHAR (30) INLINE	The name of the log hold up to which this checkpoint purged log records. For example: Backup, Checkpoint, or Oldest Transaction Undo.
additional_details	TT_VARCHAR (1000)	Additional information provided for this checkpoint, such as error codes and timestamps for multiple failed checkpoints.

Examples

This example shows a checkpoint in progress:

```
< 2019-02-05 16:56:34.169520, <NULL>,
Fuzzy          , In Progress    , User           ,
BuiltIn        , <NULL>,
0, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>,
<NULL>, <NULL>, <NULL>, 13, 6, 0, <NULL>, <NULL> >

< 2019-02-05 16:55:47.703199, 2019-02-05 16:55:48.188764,
Fuzzy          , Completed      , Checkpointer   ,
Background     , <NULL>,
1, 0, 8964304, 294, 33554432, 291, 5677288, 27, 1019512,
1065408, <NULL>, 5, 0, Checkpoint, <NULL> >

< 2019-02-05 16:54:47.106110, 2019-02-05 16:54:47.723379,
Static         , Completed      , Subdaemon      ,
FinalCkpt      , <NULL>,
0, 0, 8960328, 294, 33554432, 291, 5677288, 256, 33157172,
5321548, <NULL>, 4, 0, Checkpoint, <NULL> >

< 2019-02-05 16:54:41.633792, 2019-02-05 16:54:42.568469,
Blocking       , Completed      , User           ,
BuiltIn        , <NULL>,
1, 0, 8958160, 294, 33554432, 291, 5677288, 31, 1162112,
6604976, <NULL>, 3, 0, Checkpoint, <NULL> >

< 2019-02-05 16:54:37.438827, 2019-02-05 16:54:37.977301,
```

```
Static         , Completed     , User           ,
DbCreate       , <NULL>,
0, 0, 1611984, 93, 33554432, 92, 1853848, 93, 33554432,
1854052, <NULL>, 2, 0, Checkpoint, <NULL> >

< 2019-02-05 16:54:36.861728, 2019-02-05 16:54:37.438376,
Static         , Completed     , User           ,
DbCreate       , <NULL>,
1, 0, 1609936, 93, 33554432, 92, 1853848, 93, 33554432,
1854052, <NULL>, 1, 0, Checkpoint, <NULL> >
```

This example shows that an error occurred during the most recent checkpoint attempt, which was a user-initiated checkpoint:

```
< 2019-02-05 16:57:14.476860, 2019-02-05 16:57:14.477957,
Fuzzy          , Failed , User           ,
BuiltIn        , 847,
1, <NULL>, <NULL>, 0, 0, 0, 0, 0, 0, 0, <NULL>, 7, 0, <NULL>,
Errors  1: TT0847: 16:57:14 (2019-02-05) >

< 2019-02-05 16:56:34.169520, 2019-02-05 16:56:59.715451,
Fuzzy          , Completed     , User           ,
BuiltIn        , <NULL>,
0, 0, 8966472, 294, 33554432, 291, 5677288, 5, 522000,
532928, <NULL>, 6, 0, Checkpoint, <NULL> >

< 2019-02-05 16:55:47.703199, 2019-02-05 16:55:48.188764,
Fuzzy          , Completed     , Checkpointer   ,
Background     , <NULL>,
1, 0, 8964304, 294, 33554432, 291, 5677288, 27, 1019512,
1065408, <NULL>, 5, 0, Checkpoint, <NULL> >

< 2019-02-05 16:54:47.106110, 2019-02-05 16:54:47.723379,
Static         , Completed     , Subdaemon      ,
FinalCkpt      , <NULL>,
0, 0, 8960328, 294, 33554432, 291, 5677288, 256, 33157172,
5321548, <NULL>, 4, 0, Checkpoint, <NULL> >

< 2019-02-05 16:54:41.633792, 2019-02-05 16:54:42.568469,
Blocking       , Completed     , User           ,
BuiltIn        , <NULL>,
1, 0, 8958160, 294, 33554432, 291, 5677288, 31, 1162112,
6604976, <NULL>, 3, 0, Checkpoint, <NULL> >

< 2019-02-05 16:54:37.438827, 2019-02-05 16:54:37.977301,
Static         , Completed     , User           ,
DbCreate       , <NULL>,
0, 0, 1611984, 93, 33554432, 92, 1853848, 93, 33554432,
1854052, <NULL>, 2, 0, Checkpoint, <NULL> >

< 2019-02-05 16:54:36.861728, 2019-02-05 16:54:37.438376,
Static         , Completed     , User           ,
DbCreate       , <NULL>,
1, 0, 1609936, 93, 33554432, 92, 1853848, 93, 33554432,
1854052, <NULL>, 1, 0, Checkpoint, <NULL> >
```

This example selects specific columns from the checkpoint history:

```
select type, reason, bookmarkname, logsPurged from ttCkptHistory;
< Fuzzy          , BuiltIn        , Oldest Transaction Undo, 0 >
< Static         , FinalCkpt      , Checkpoint, 6 >
```

```
< Blocking       , BuiltIn       , Checkpoint, 0 >
< Blocking       , BuiltIn       , Checkpoint, 0 >
< Blocking       , BuiltIn       , Checkpoint, 0 >
< Blocking       , BuiltIn       , Backup, 5 >
< Blocking       , BuiltIn       , Backup, 0 >
< Blocking       , BuiltIn       , Backup, 0 >
```

The bottom (oldest) checkpoints could not purge log files because there was a log hold set by incremental backup, but eventually the log hold moved and five log files could be purged.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure returns a row for the element from which it was called. To see information about other elements, query the `SYS.GV$COMMIT_BUFFER_STATS` system table.

Related views

This procedure has these related views.

`SYS.GV$COMMIT_BUFFER_STATS`

`SYS.V$COMMIT_BUFFER_STATS`

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( );
< 10, 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has these related views.

`SYS.GV$CONFIGURATION`

`SYS.V$CONFIGURATION`

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`.'

Required privilege

This procedure requires the `ADMIN` privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

Syntax

```
ttCompact()
```

Parameters

`ttCompact` has no parameters.

Result set

`ttCompact` returns no results.

Examples

```
CALL ttCompact;
```

Notes

Compacting data does not modify result addresses.

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, dictionary tables, 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

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

Parameter	Type	Description
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 allows 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 and some options for the current database connection.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$CONFIGURATION

SYS.V$CONFIGURATION

Syntax

```
ttConfiguration(['paramName'])
```

Parameters

`ttConfiguration` has the optional parameter:

Parameter	Type	Description
paramName	TT_VARCHAR (30)	The name of a connection attribute or option 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();
< CkptFrequency, 600 >
< CkptLogVolume, 0 >
. . .
```

Notes

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

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

See also

Chapter 2, "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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$CONTEXT

SYS.V$CONTEXT

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

"ttStatus" on page 5-165

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$DATASTORE_STATUS

SYS.V$DATASTORE_STATUS

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.

Column	Type	Description
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 5-165

ttDBCompactConfig

Description

The ttDBCompactConfig built-in procedure turns on automatic database compaction. By default, TimesTen does not compact databases automatically.

Required privilege

This procedure requires ADMIN privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$DB_COMPACT_CONFIG

SYS.V$DB_COMPACT_CONFIG

Syntax

ttDBCompactConfig([[*value*][,[*value*]][,[*value*]]])

Parameters

ttDBCompactConfig has the parameters:

Parameter	Type	Description
quantum	TT_INTEGER	Specifies the number of data blocks to be compacted. Values from 0 to 100000 are allowed. A value of 0 means that automatic database compaction is disabled.
compactsPerSecond	TT_INTEGER	Number of compaction operations that can occur per second. Values from 0 to 100 are allowed. A value of 0 means that automatic database compaction is disabled.
threshold	TT_INTEGER	Specifies the minimum amount of the database that needs to be compacted, before automatic database compaction occurs. The units are the same as for parameter quantum. Values from 0 to 100000 (100k) are allowed. A value of 0 means that the compaction operations executes whenever there is anything to compact in the database.

Result set

ttDBCompactConfig returns the result set:

Column	Type	Description
quantum	TT_INTEGER NOT NULL	Currently active setting for quantum.
compactsPerSecond	TT_INTEGER NOT NULL	Currently active setting for compactsPerSecond.
threshold	TT_INTEGER NOT NULL	Currently active setting for threshold.

Examples

To view the current settings for automatic database compaction, use:

```
Call ttDbCompactConfig;
< 0,0,0 >
1 row found.
```

Note: These are the default settings. Automatic database compaction is disabled if either of the first two parameters is 0.

To enable automatic database compaction on 1,000 blocks once a second, use:

```
Call ttDbCompactConfig (1000,1,0);
< 1000,1,0 >
1 row found.
```

To enable automatic database compaction on 5,000 blocks ten times a second, use:

```
Call ttDbCompactConfig (5000,10,0);
< 5000,10,0 >
1 row found.
```

To enable automatic database compaction on 2,000 blocks five times a second, but only perform compaction when there is at least this much to compact, use:

```
Call ttDbCompactConfig (2000,5,2000);
< 2000,5,2000 >
1 row found.
```

To stop automatic database compaction from doing further compaction, after it was turned on (note that it is off by default), use:

```
Call ttDbCompactConfig (0,0,0);
```

You can set just one or two values at a time, using commas as placeholders:

```
call ttDBCompactConfig(2000,5,2000);
< 2000, 5, 2000 >
1 row found.
call ttDBCompactConfig(3000);
< 3000, 5, 2000 >
1 row found.
call ttDBCompactConfig(,10);
< 3000, 10, 2000 >
1 row found.
call ttDBCompactConfig(,,2500);
< 3000, 10, 2500 >
1 row found.
call ttDBCompactConfig(3500,,3000);
< 3500, 10, 3000 >
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 `ttCkpt` 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.

You can specify one, two, or three input values, using commas as placeholders, or no input to see the current values.

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

See Also

ttDBConfig

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

`SYS.GV$DB_CONFIG`

`SYS.V$DB_CONFIG`

Syntax

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

Parameters

`ttDBConfig` has the parameters:

Parameter	Type	Description
param	VARCHAR2(30)	.A system parameter for which you either want to set a value or see the current value. Accepted values for this argument are:
		`CacheAgentCommitBufSize`
		`CacheAwtMethod`
		`CacheParAwtBatchSize`
		`DynamicLoadReduceContention`
		`ParReplMaxDrift`
		`PLSQL_OPEN_CURSORS`
		`RepAgentCommitBufSize`
value	VARCHAR2(200)	The value of 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
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*.
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 connection attribute value.
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.
DynamicLoadReduceContention	0 - Disabled. Default. 1 - Enabled.	If enabled, changes the way that autorefresh and dynamic load operations coordinate, which results in reduced contention between autorefresh and dynamic load operations. ■ Dynamic load operations are never blocked by autorefresh operations (due to additional synchronization). ■ Autorefresh operations are not completely delayed by dynamic load operations. For more details, see "Reducing contention on TimesTen for dynamic read-only cache groups with incremental autorefresh" in the *Oracle TimesTen Application-Tier Database Cache User's Guide*.

Name	Value	Description
ParReplMaxDrift	Number of seconds	Specifies the number of seconds of drift to allow between the parallel replication tracks. When you use automatic parallel replication with disabled commit dependencies, some of the 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.
PLSQL_OPEN_CURSORS	Maximum number of PL/SQL cursors	Specifies the maximum number of PL/SQL cursors that can be open in a session at one time, a value from 1 to 65535, inclusive. Use this to prevent a session from opening an excessive number of cursors. Default is 50 PL/SQL cursors.
		For more details see the description of the PLSQL_OPEN_CURSORS first connection attribute.
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 specified parameter.
value	VARCHAR2(200)	The current value of the specified parameter. This is the new value, if you specified a value.

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.
```

The following example sets DynamicLoadReduceContention=1:

```
call ttDbConfig('DynamicLoadReduceContention','1');
```

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 ttCkpt 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.

If you call ttDBConfig without an input parameter, it will return names and values of all supported parameters.

See also

"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 transactions.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has these related views.

`SYS.GV$DB_WRTE_CONCURRENCY_MODE`

`SYS.V$DB_WRTE_CONCURRENCY_MODE`

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

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

ttDistributionProgress

Description

This built-in procedure provides a progress report of an ongoing redistribution process.

Required privilege

This procedure requires the ADMIN privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is not supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

This procedure returns a row for the element from which it was called.

Related views

This procedure has no related views.

Syntax

```
ttDistributionProgress()
```

Parameters

ttDistributionProgress has no parameters.

Result set

ttDistributionProgress returns the result:

Column	Type	Description
timestamp	TT_TIMESTAMP	Time at which the status information was collected.
elementId	TT_INTEGER	Element ID of the local element.
ptVersion	TT_INTEGER	Version number of the partition table.
threadId	TT_INTEGER	Thread ID.
phase	VARCHAR2 (32)	Current phase of the redistribution process. The redistribution process has the following phases: ■ Data Distribution ■ Data Checkpoint ■ Data Checkpoint Done ■ Reclaim Phase 1 ■ Reclaim Checkpoint ■ Reclaim Phase 2 ■ Reclaim Done
tblName	VARCHAR2 (64)	Name of the table currently being processed, if available.

Column	Type	Description
processedTblRows	TT_BIGINI	Number of rows already processed of the current table, if available.
insertedTblRows	TT_BIGINT	Number of rows already inserted to the current table in the local element, if available.
deletedTblRows	TT_BIGINT	Number of rows already deleted from the current table in the local element, if available.
totalTblRows	TT_BIGINT	Total number of rows in the table, if available.
processedRows	TT_BIGINT	Number of rows already processed for the element.
insertedRows	TT_BIGINT	Number of rows already inserted to the element.
deletedRows	TT_BIGINT	Number of rows already deleted from the element.
totalRows	TT_BIGINT	Total number of rows in the element for all tables.
processedTbls	TT_INTEGER	Number of tables already processed.
totalTbls	TT_INTEGER	Total number of tables in the database.

Examples

The following example shows an example result set for a call to the
ttDistributionProgress built-in procedure.

```
Command> CALL ttDistributionProgress();
< 2018-12-04 14:49:41.065122, 1, 2, 1, Data Distribution, <NULL>, <NULL>, <NULL>,
 <NULL>, <NULL>, 1910, 0, 176, 1910, 8, 8 >
1 row found.
```

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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).

ttEpochCreate

Description

This procedure causes the next committed transaction to commit as an epoch transaction. An epoch is a transaction that marks a globally consistent point in time across all elements of the database. An epoch is durably committed in every replica set of a grid.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is not supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Restrictions with TimesTen Classic

TimesTen Classic applications cannot call this built-in procedure.

Related views

This procedure has no related views.

Syntax

```
ttEpochCreate()
```

Parameters

ttEpochCreate has no parameters.

Result set

ttEpochCreate returns no results.

Examples

```
CALL ttEpochCreate;
```

ttEpochSessionGet

Description

This procedure returns the epoch identifier of the last epoch created by the current connection, if one is available.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is not supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Restrictions with TimesTen Classic

TimesTen Classic applications cannot call this built-in procedure.

Related views

This procedure has these related views.

```
SYS.GV$EPOCH_SESSION
```

```
SYS.V$EPOCH_SESSION
```

Syntax

```
ttEpochSessionGet()
```

Parameters

ttEpochSessionGet has no parameters.

Result set

ttEpochSessionGet returns the result set:

Column	Type	Description
epoch	TT_VARCHAR (50)	The epoch session ID, if available.

Examples

```
CALL ttEpochSessionGet();
< 1023 >
```

Notes

ttHeapInfo

Description

This procedure reports heap memory usage in the database. For each heap in the database, it displays the allocated size, size in use, high water mark (the maximum amount of size in use) and the number of deferred free buffers.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$HEAP_INFO

SYS.V$HEAP_INFO

Syntax

ttHeapInfo('*name*')

Result set

ttHeapInfo has the parameter:

Column	Type	Description
name	TT_CHAR (30)	Name of the heap for which you would like information

Result set

ttHeapInfo returns the result:

Column	Type	Description
name	TT_CHAR (30) NOT NULL	Name of the database for which the heap memory info is being returned.
size	TT_BIGINT NOT NULL	The allocated sizes of the heap memory.
inUse	TT_BIGINT NOT NULL	The amount of heap memory in use.
highWater	TT_BIGINT NOT NULL	The maximum amount of heap memory used.
freeDeferred	TT_BIGINT NOT NULL	The number of deferred freed heap memory buffers.

Examples

```
CALL ttHeapInfo ('sampledb1');
< PERMANENT_0, 2515656, 2404112, 2582856, 0>
< PERMANENT_1, 1024, 1024, 1024, 0>
...
< INDEX_SNAPSHOT_VALUE_CONFIG_I, 2048, 776, 776, 0>
156 rows found
```

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

This procedure returns a row for the element from which it was called.

Related views

This procedure has these related views.

SYS.GV$HOST_NAME

SYS.V$HOST_NAME

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 ();
< myhost >
1 row found.
```

See also

ttHostNameSet

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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.

Columns	Type	Description
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.
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, 2018-02-27 11:44:08.136833,
2018-02-27 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$INDEX_ADVICE_OUTPUT

SYS.V$INDEX_ADVICE_OUTPUT

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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*

ttLatchStatsGet

Description

This procedure displays latch statistics. Statistics are useful for determining areas of contention in a running system.

This procedure is primarily meant to be used when requested by TimesTen technical support.

Required privilege

This procedure requires `ADMIN` privileges to show all active connections or database level statistics. No privileges are required to show the current connection's latch statistics.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

Syntax

```
ttLatchStatsGet (level, operation)
```

Parameters

`ttLatchStatsGet` has these parameters:

Parameter	Type	Description
level	TT_CHAR (16)	The level controls the number of connections for which the stats are printed. Valid values are:
		`db` - All the active connections on the database.
		`conn` - The current connection. This is the default.
		`connid` - An specific connection (specified by connid).
operation	TT_CHAR (16)	This value controls the verbosity level of the output. Valid values are:
		`show` - Only show the contention points that have a high contention level
		`showall` - Show the contention points that have contention
		`showallandtell` - Show all the contention points

Result set

Results sets are are divided into two types: contention point and statistics.

ttLatchStatsGet returns the result set for contention points. These describe the location of contention.

Column	Type	Description
callerName	TT_VARCHAR(40) NOT NULL	Function name containing the contention point.
fileName	TT_VARCHAR(30) NOT NULL	The file that contains the *callerName* function.
lineNo	TT_INTEGER NOT NULL	The line number of the *fileName* file.
description	TT_VARCHAR(100) NOT NULL	Description of this contention point.

ttLatchStatsGet returns the result set for statistics. These describe detailed statistics about this contention point:

Column	Type	Description
connName	TT_VARCHAR(40) NOT NULL	The name of the connection experiencing contention.
spinCount	TT_BIGINT NOT NULL	The number of times the *connName* connection has spun on this contention point.
access	TT_BIGINT NOT NULL	The number of times the *connName* connection has used this contention point.
sleepCnt	TT_INTEGER NOT NULL	The number of times the *connName* connection has slept on this contention point.
firstTry	TT_INTEGER NOT NULL	The number of times the *connName* connection has used this contention point without experiencing contention.
collisions	TT_INTEGER NOT NULL	The number of times the *connName* connection has used this contention point and experienced contention.
avgSpin	TT_BIGINT NOT NULL	The average number of times the *connName* connection has spun on this contention point.

Examples

The following example shows an example result set for a call to ttLatchStatsGet.

```
Command> CALL ttLatchStatsGet
< getSmallMed, heap.c 2675, Generic description, sampledb1,
  1, 0, 1, 0, -1, -1, 0 >
< sbhpallocAttempt, heap.c 3712, Generic description,
  sampledb1, 1, 0, 1, 0, -1, -1, 0 >
```

See also

"ttXactAdmin" on page 5-176

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. While performing the load, an implicit commit is executed after every 256 rows inserted into the TimesTen database.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

Syntax

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

Parameters

ttLoadFromOracle has these parameters:

Parameter	Type	Description
tblOwner	TT_CHAR (30)	TimesTen table owner (optional). If not provided, the connection ID is used.

Parameter	Type	Description
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.
SelectSQL	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.
numThreads	TT_INTEGER	Number of threads for parallel load (optional). If NULL, defaults to 4.
		Provides parallel loading for tables. Specifies the number of loading threads to run concurrently. One thread performs the bulk fetch from the Oracle database 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.
		You can also use the readers option to specify the total number of threads from the numThreads parameter to use for bulk fetching from the Oracle database.
Options	TT_VARCHAR (1000)	Option string, specified as key=value pairs. For legal values, see "Options string".
		Defaults to NULL.
		See the table below for more information.

Options string

The options are specified as key = value pairs and the pairs are separated by semi-colons.

Option parameter	Value	Description
localOnly	Y or N	This option only loads rows from a specific instance. Load a specific instance in the grid and use this option. When you use this option, ttLoadFromOracle selects all rows from the table, but ignores any rows that are not hashed to the specific instance.
		This option is only supported in TimesTen Scaleout.
		The default value is N.

Option parameter	Value	Description
ErrorThreshold	value > 0	This option sets the error threshold for which the built-in procedure returns an error messages.
		The default value is 1. When the ttLoadFromOracle built-in procedure encounters an error, the built-in procedure stops and returns an error message.
IgnoreDuplicates	Y or N	This option makes the ttLoadFromOracle built-in procedure ignore uniqueness constraint violations, which results in duplicates being ignored. You can only use this option if the TimesTen table has a uniqueness constraint on it.
		The default value is N.
ResumeFromSCN	*scn_of_last_load*	This option resumes the load operation from the specific SCN. When the resumeFromSCN option is enabled, it automatically ignores duplicates.
		The default value is the latest SCN.
DirectLoad	Y or N	This option enables a bulk insert mode which has performance benefits.
		This option is only supported in TimesTen IMDB.
		The default value is N.
readers	numThreads > value > 0	This option specifies the total number of threads from the numThreads parameter to use for bulk fetching from the Oracle database.
		For example, if you specify a numThreads parameter of 8 and a readers option of 3, 3 threads bulk fetch data from the Oracle database and 5 threads load data into the TimesTen database.

Result set

ttLoadFromOracle returns the result set:

Column	Type	Description
numRows	TT_BIGINT	Number of rows loaded.
numErrors	TT_BIGINT	Number of rows with errors.
errCode	TT_INTEGER	TimesTen error code, one of:
		0 - Load completed successfully without errors
		-1 - Load completed successfully with errors
		-2 - Load terminated early with errors
		-3 - Load terminated early with a fatal error, for example, an out-of-space error, a loss of connection or an invalidation.

Column	Type	Description
errMsg	TT_VARCHAR(4000)	Error message, containing:
		■ Start and end time of load
		■ Statement
		■ SCN used to query the data
		■ Number of rows with errors
		■ Number of rows loaded

Examples

The following example selects loads the TimesTen table about employees from the Oracle database HR.EMPLOYEES table and loads it into the TimesTen HR.EMPLOYEES table. In this example an error is returned.In this example, the column STATE is a TT_TINYINT.

```
Command> CALL ttLoadFromOracle ('HR','EMPLOYEES',
'SELECT * FROM HR.EMPLOYEES');
< 99, 0, 0, 'Started=2014-08-01 13:48:21; Ended=2014-08-01 13:48:23;
 Statement=ttLoadFromOracle('HR', 'SELECT * FROM HR.EMPLOYEES'); SCN=1234567;
 Errors=1; Rows Loaded=99' >
<  NULL, NULL, 2614,'Value outside of range supported by integral type. Column
 STATE=-1' >
```

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 2-64 for full details of the different locking levels.

Required privilege

This procedure requires the ADMIN privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

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 2-64

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 records" 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

This procedure returns a row for the element from which it was called. To see information about other elements, query the SYS.GV$LOG_HOLDS system table.

Related views

This procedure has these related views.

SYS.GV$LOG_HOLDS

SYS.V$LOG_HOLDS

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.

Column	Type	Description
type	TT_CHAR (30) NOT NULL	Returns the type of hold, one of: Checkpoint Replication Backup XLA Long-Running Transaction Long-Running XA Transaction TTGrid Replica Element Duplicate
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 ■ The index of the replica in the partition table, the replica id, the index of the local element, the version of partition table for the replica log hold, and the index for the loop of the list of replicas. ■ The string Log hold of Element Duplicate used by LBCU.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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 owned by the current user if *tblName* is not specified. If the user is the instance administrator, only tables owned by the instance administrator are updated. If the tables are not owned by the user, the user can qualify the table name with their own user name to update stats for the current user.

To determine if your stats are updated, look at the system tables, SYS.COL_STATS and SYS.TBL_STATS, before and after you perform this operation.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has these related views.

SYS.GV$OPT_COL_STATS

SYS.V$OPT_COL_STATS

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.

Parameter	Type	Description
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 large or the table in question is 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$OPT_COL_STATS

SYS.V$OPT_COL_STATS

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.

Column	Type	Description
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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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 3-171 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$OPT_MAX_CMD_FREELIST_CNT

SYS.V$OPT_MAX_CMD_FREELIST_CNT

Syntax

ttOptGetMaxCmdFreeListCnt()

Parameters

ttOptGetMaxCmdFreeListCnt has no parameters.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is supported in TimesTen Scaleout.

This procedure returns a row for the element from which it was called. To see information about other elements, query the SYS.GV$OPT_ORDER system table.

Related views

This procedure has these related views.

SYS.GV$OPT_ORDER

SYS.V$OPT_ORDER

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).

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

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).

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

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).

Parameter	Type	Description
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 NULLs 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 resets all optimizer flags to their default values when the transaction has been committed or rolled back. This alters 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. If optimizer flags are set while `AutoCommit` is on, they are ignored.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

`SYS.GV$OPT_FLAG`

`SYS.V$OPT_FLAG`

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.
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*.
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.
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.
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.
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, "Summary of statement, transaction, and connection level optimizer hints" in the Oracle TimesTen In-Memory Database SQL Reference 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

Syntax

```
ttOptSetMaxPriCmdFreeListCnt(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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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, "Summary of statement,

transaction, and connection level optimizer hints" in the *Oracle TimesTen In-Memory Database SQL Reference* 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).

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$OPT_JOIN_ORDER

SYS.V$OPT_JOIN_ORDER

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

```
Command> 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
ttOptGetOrderd
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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Grid applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$OPT_STATS

SYS.V$OPT_STATS

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 owned by the current user if *tblName* is not specified. If the user is the instance administrator, only tables owned by the instance administrator are updated. If the tables are not owned by the user, the user can qualify the table name with their own user name to update stats for the current user.

To determine if your stats are updated, look at the system tables, SYS.COL_STATS and SYS.TBL_STATS, before and after you perform this operation.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes on all elements in the grid.

Related views

This procedure has no related views.

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.

Parameter	Type	Description
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.
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 3-157 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

This procedure returns a row for the element from which it was called. To see information about other elements, query the SYS.GV$PLSQL_MEMORY_STATS system table.

Related views

This procedure has these related views.

SYS.GV$PLSQL_MEMORY_STATS

SYS.V$PLSQL_MEMORY_STATS

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;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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttRamPolicyAutoReloadGet()
```

Parameters

`ttRamPolicyAutoReloadGet` has no parameters.

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.

Examples

To view the RAM autoreload policy, use:

```
CALL ttRamPolicyAutoReloadGet();
```

See also

ttRamPolicyAutoReloadSet
"ttAdmin" on page 5-4

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 5-4

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttRamPolicyGet()
```

Parameters

ttRamPolicyGet has no parameters.

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 (default) - 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. This policy is not supported in TimesTen Scaleout.
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 5-4
"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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 5-4
"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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$REDUNDANT_INDEX

SYS.V$REDUNDANT_INDEX

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 "Set 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 daemon log. 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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 daemon log. 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Examples

To determine the replication state of the active standby pair, use:

```
Call ttRepStateGet();
<STANDBY>

Call ttRepStateGet();
```

```
<ACTIVE>

Call ttRepStateGet();
<FAILED>
```

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Parameter	Type	Description
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.") "Set 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.

Procedure is working expected for transient error scenarios.

> **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.
>
> The procedure does not return any failure output (01 value) for permanent error scenarios.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.
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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttRepSyncGet()
```

Parameters

ttRepSyncGet has no parameters.

Result set

ttRepSyncGet returns the result set:

Column	Type	Description
requestReturn	BINARY(1)	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.

Column	Type	Description
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
                   , /* Tarype */       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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Parameter	Type	Description
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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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*.
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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttRepXactTokenGet('typ')
```

Parameters

ttRepXactTokenGet has these parameters:

Parameter	Type	Description
typ	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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

This procedure returns a row for the element from which it was called. To see information about other elements, query the SYS.GV$TABLE_SIZES system table.

Related views

This procedure has no related views.

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 optional. If not specified all table sizes are returned. Using a synonym to specify a table name is not supported.

Parameter	Type	Description
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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$SQL_CMD_CACHE

SYS.V$SQL_CMD_CACHE

Syntax

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

Parameters

ttSQLCmdCacheInfo has the optional parameter:

Parameter	Type	Description
sqlCmdID	TT_BIGINT	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_BIGINT NOT NULL	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 -1.
executions	TT_BIGINT NOT NULL	Counts the number of executions of the command.

Column	Type	Description
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.
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 NOT NULL	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.
execloc	TT_TINYINT NOT NULL	
gridCmdId	TT_VARCHAR (64)	
tempSpaceUsage	TT_BIGINT	
maxTempSpaceUsage	TT_BIGINT	

Examples

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

```
Command> call ttsqlcmdcacheinfo;
< 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 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#=:
  FETCHCOUNT:
  STARTTIME:
  MAXEXECUTETIME:
  LASTEXECUTETIME:
  MINEXECUTETIME:

1 row found.
```

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$SQL_CMD_CACHE_INFO

SYS.V$SQL_CMD_CACHE_INFO

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

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$SQL_CMD_QUERY_PLAN

SYS.V$SQL_CMD_QUERY_PLAN

Syntax

ttSQLCmdQueryPlan([*sqlCmdID*])

Parameters

ttSQLCmdQueryPlan has the optional parameter:

Parameter	Type	Description
sqlCmdID	TT_BIGINT	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_BIGINT NOT NULL	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.

Column	Type	Description
level	TT_INTEGER	The level number of current operation in this run-time query plan.
operation	TT_CHAR(127)	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.
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.
miscellaneous	TT_VARCHAR (65536)	The type of constraint and other information about the constraint. Constraint type can be one of:
		ForeignKeyInsert - To insert foreign key.
		ForeignKeyDelete - To delete foreign key.
		UniqueKeyInsert - To insert unique key.
		ForeignKeyOrphanChild - To handle case where parent is lost in case of foreign key constraint.
		ForeignKeyCascadeDelete - To delete corresponding row for cascade delete.
		ForeignKeySyncCascadeDelete - To delete corresponding row for cascade delete from sync replica.

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> >
< 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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

`SYS.GV$EXECUTION_TIME_HISTOGRAM`

`SYS.V$EXECUTION_TIME_HISTOGRAM`

Syntax

```
ttSQLExecutionTimeHistogram(sqlCmdID)
```

Parameters

`ttSQLExecutionTimeHistogram` has the optional parameter:

Parameter	Type	Description
`sqlCmdID`	`TT_BIGINT`	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.

Column	Type	Description
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 0.00001562 seconds.

- 1484 statements executed in a time frame that was greater than 0.00001562 seconds and less than or equal to 0.000125 seconds.

- 35 statements executed in a time frame that was greater than 0.000125 seconds and less than or equal to 0.001 seconds.

- 62 statements executed in a time frame that was greater than 0.001 seconds and less than or equal to 0.008 seconds.

- 60 statements executed in a time frame that was greater than 0.008 seconds and less than or equal to 0.064 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

ttStatsConfig

Description

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

Required privilege

This procedure requires the `ADMIN` privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is supported in TimesTen Scaleout, but supports different parameter/value pairs than in TimesTen Classic.

In TimesTen Scaleout, this procedure broadcasts changes to all elements.

Related views

This procedure has no related views.

Syntax

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

Parameters

`ttStatsConfig` has the parameters:

Parameter	Type	Description
param	VARCHAR2(50) NOT NULL	The name of the parameter to configure.
value	VARCHAR2(200)	The value of the specified parameter. If no value is supplied, the built-in procedure displays the current value for the specified parameter.
option	VARCHAR2 (200)	

Parameter / Value Pairs

The supported parameter/value pairs in TimesTen Classic and TimesTen Scaleout are different. These are the supported parameter/value pairs:

- TimesTen Classic
- TimesTen Scaleout

TimesTen Classic

These parameter/value pairs can be set with TimesTen Classic:

Parameter	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.
LatchStats	scope,level scope=conn|db|*con_id* level=NONE|TYPICAL| ALL|BASIC	Specifies the scope, *scope*, and the level, *level*, for collection for latch statistics. The scope value determines at what level TimesTen should collect latch statistics: ■ conn - Collects latch statistics for your current connection. ■ db - Collects latch statistics for your database. ■ *con_id* - Collects latch statistics for the connection name that you specify. The level value determines the level at which TimesTen collects statistics: ■ NONE - Disables the collection of latch statistics. ■ TYPICAL - Ensures the collection of major useful latch statistics. ■ ALL - Additional statistics are added to the set of statistics collected with the TYPICAL setting. The additional statistics include internal and debugging statistics. ■ BASIC - Disables the collection of many of the important latch statistics. If latch statistics are enabled, TimesTen allocates around 100KB from temporary memory to store these statistics. Once you have configured the LatchStats parameter, you can use the ttLatchStatsGet built-in procedure to view latch statistics. See "ttLatchStatsGet" on page 3-139 for more information.
SQLCmdHistogramReset	0 or not	The existing SQL execution time statistics are reset if the specified value is nonzero.
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.

Parameter	Value	Description
StatsLevel	NONE TYPICAL ALL BASIC	Specifies the level of collection for database statistics. TimesTen stores theses statistics in system tables. Setting the StatsLevel parameter to NONE disables the collection of system statistics. The default setting of TYPICAL ensures collection of major useful statistics and 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 include internal and debugging statistics. Setting the StatsLevel parameter to BASIC disables the collection of many of the important statistics required by many TimesTen features.

TimesTen Scaleout

These parameter/value pairs can be set with TimesTen Scaleout:

Parameter	Value	Description
pollSec	0 10 <= *value* <= 60	The polling interval, in seconds, at which the ttStats daemon captures snapshots of the TimesTen Scaleout. A value of 0 disables the ttStats daemon from capturing metrics. The value of the polling interval does not affect the performance of the TimesTen Scaleout. However, a polling interval of 60 seconds tends to use six times more space than a polling interval of 10 seconds. Ensure that you have sufficient PermSize to support the desired polling interval. The default value is 30 seconds.
retainMinutes	15<= *value* <= 1440	The time, in minutes, that the ttStats daemon waits before aggregating and purging raw metrics. If you use a larger value for retainMinutes, the ttStats daemon stores more metrics in the system tables. The default value is 120 minutes.
retentionDays	1 < *value* < 730	The retention time interval, in days, at which the ttStats daemon drops ttStats snapshots of the TimesTen Scaleout. For example, if the retention time interval is 62 days, the ttStats daemon drops the 1st day's snapshot on the 63rd day. Ensure that you have sufficient PermSize to support the desired retention time interval. In most cases, a day's worth of data takes up around 20 MB of space. These metrics are stored in SYS tables and survive database bounces. The default value is 62 days.

Result set

ttStatsConfig returns the result set:

Column	Type	Description
param	VARCHAR2(50) NOT NULL	The name of the parameter that was configured.
value	VARCHAR2(200)	The value of the specified parameter. If no value is supplied, the built-in procedure displays the current value for the specified parameter.

Examples

Since TimesTen and TimesTen Scaleout support different parameter/value pairs, there are also different examples. These are supported examples:

- TimesTen Classic
- TimesTen Scaleout

TimesTen Classic

Sample every command:

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

Check 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.
```

TimesTen Scaleout

Sets the polling interval of statistics to 45 seconds. Therefore, the ttStats daemon aggregates statics every 45 seconds:

```
Command> call ttStatsConfig('pollsec', 45);
< POLLSEC, 45 >
1 row found.
```

Sets the time interval when the `ttStats` daemon purges raw metrics to 60 minutes:

```
Command> call ttStatsConfig('retainMinutes', 60);
< RETAINMINUTES, 60 >
1 row found.
```

Sets the retention time interval for statistics to 30 days:

```
Command> call ttStatsConfig('retentionDays', 30);
< RETENTIONDAYS, 30 >
1 row found.
```

See also

"ttStats" on page 5-137
ttStatsConfigGet

ttStatsConfigGet

Description

The `ttStatsConfigGet` built-in procedure returns parameters of the `ttStats` utility that you can set with the `ttStatsConfig` built-in procedure. This procedure does not take any input and outputs a multiple row result set with name/value pair parameters.

Required privilege

This procedure requires the `ADMIN` privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is supported in TimesTen Scaleout, but supports different parameter/value pairs than in TimesTen Classic.

In TimesTen Scaleout, this procedure returns a row for the element from which it was called. To see information about other elements, query the `SYS.GV$STATS_CONFIG` system table.

Related views

This procedure has these related views.

`SYS.GV$STATS_CONFIG`

`SYS.V$STATS_CONFIG`

Syntax

```
ttStatsConfigGet()
```

Parameters

`ttStatsConfigGet` has no parameters:

Result set

`ttStatsConfigGet` returns the result set:

Column	Type	Description
param	VARCHAR2(50)	The name of the parameter.
value	VARCHAR2(200)	The current value of the parameter.

Parameter / Value Pairs

The supported return parameter/value pairs in TimesTen and TimesTen Scaleout are different. These are the return parameter/value pairs:

- TimesTen Classic
- TimesTen Scaleout

TimesTen Classic

These parameter/value pairs can be returned in the result set in TimesTen Classic:

Parameter	Description
SQLCmdSampleFactor	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.
ConnSampleFactor	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.
StatsLevel	The existing SQL execution time statistics are reset if the specified value is nonzero.

TimesTen Scaleout

These parameter/value pairs can be returned in the result set in TimesTen Scaleout:

Parameter	Description
pollSec	The polling interval, in seconds, at which the ttStats daemon captures snapshots of the TimesTen Scaleout. A value of 0 disables the ttStats daemon from capturing metrics.
retainMinutes	The time, in minutes, that the ttStats daemon waits before aggregating and purging raw metrics.
retentionDays	The retention time interval, in days, at which the ttStats daemon drops ttStats snapshots of the TimesTen Scaleout. For example, if the retention time interval is 62 days, the ttStats daemon drops the 1st day's snapshot on the 63rd day.

Examples

Since TimesTen and TimesTen Scaleout support different name/value pair results, there are also different examples. These are supported examples:

- TimesTen Classic

- TimesTen Scaleout

TimesTen

View the configuration settings of ttStatsConfig:

```
Command> call ttStatsConfigGet();
< SQLCMDSAMPLEFACTOR, 1 >
< CONNSAMPLEFACTOR, 2047,0 >
< STATSLEVEL, TYPICAL >
3 rows found.
```

TimesTen Scaleout

View the configuration settings of `ttStatsConfig`:

```
Command> call ttStatsConfigGet();
< POLLSEC, 10 >
< RETAINMINUTES, 120 >
< RETENTIONDAYS, 62 >
3 rows found.
```

See also

"ttStats" on page 5-137
ttStatsConfig

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

Syntax

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

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.
selectSQL	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 built-in procedure returns the five parts of the TimesTen release number.

Required privilege

This procedure requires no privilege.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$VERSION

SYS.V$VERSION

Syntax

```
ttVersion()
```

Parameters

ttVersion has no parameters.

Result set

ttVersion returns the result set:

Column	Type	Description
major1	TT_INTEGER NOT NULL	The first part of the five-part release number (18 for release 18.1.2.1.0), indicating the last two digits of the year of the major release. A change in major1 indicates major infrastructure and functionality changes.
major2	TT_INTEGER NOT NULL	The second part of the five-part release number (1 for release 18.1.2.1.0). A change in only major2 indicates a version with new functionality and possibly some infrastructure changes. Releases with the same major1.major2 are patch-compatible. Data can be unloaded from a database from one release and loaded into a database from the other without the migration process.

Column	Type	Description
patchset	TT_INTEGER NOT NULL	The third part of the five-part release number (2 for release 18.1.2.1.0).
		A change in only patchset indicates a release that contains bug fixes and possibly some feature enhancements.
patch	TT_INTEGER NOT NULL	The fourth part of the five-part release number (1 for release 18.1.2.1.0).
		A change in only patch indicates a release with only critical bug fixes.
reserved	TT_INTEGER NOT NULL	The fifth part of the five-part release number (0 for release 18.1.2.1.0). Reserved for future use.

Examples

Return for release 18.1.2.1.0:

```
Command>  call ttVersion();
< 18, 1, 2, 1, 0 >
1 row found.
```

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

TimesTen Scaleout applications can call this built-in procedure.

In TimesTen Scaleout, this procedure executes locally on the element from which it is called.

Related views

This procedure has these related views.

SYS.GV$XACT_ID

SYS.V$XACT_ID

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

Syntax

```
ttXlaBookmarkCreate('bookmark', 'replicated')
```

Parameters

ttXlaBookmarkCreate has the parameters:

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is supported in TimesTen Classic.

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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.

Usage in TimesTen Scaleout and TimesTen Classic

This procedure is not supported in TimesTen Scaleout.

Related views

This procedure has no related views.

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

4

TimesTen Scaleout Utilities

This chapter provides reference information for utilities that are only supported with TimesTen Scaleout, beginning with the following introductory sections:

- Overview
- Utilities list

For information about utilities that are only supported in TimesTen Classic or supported in both TimesTen Classic and TimesTen Scaleout, see Chapter 5, "Utilities".

Overview

The options for TimesTen utilities are generally not case sensitive, except for single character options. You can use `-timeout` or `-TimeOut` 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.

Utilities list

Utilities listed in Table 4–1 are described in this chapter.

Utilities listed in Table 4–2 are described in Chapter 5, "Utilities".

Table 4–1 Utilities supported only in TimesTen Scaleout descriptions

Name	Description
ttGridAdmin	Administers a TimesTen Scaleout grid.
ttGridRollout	Creates a new grid and database.

Table 4–2 Utilities supported in both TimesTen Scaleout and TimesTen Classic descriptions

Name	Description
ttInstallationCheck	Examines all files in an installation of TimesTen and generates a signature for the installation,

Table 4–2   (Cont.) Utilities supported in both TimesTen Scaleout and TimesTen Classic descriptions

Name	Description
ttInstallDSN	Generates a Windows client DSN for one or more entries listed in the provided input file and installs them into the ODBC Panel as a System DSN.
ttInstanceCreate	Create a new TimesTen instance.
ttInstanceDestroy	Destroys an existing TimesTen instance.
ttInstanceModify	Modifies certain attributes of an instance.
ttIsql	Executes SQL statements interactively.
ttMigrate	Saves and restores TimesTen objects.
ttSchema	Prints out the schema, or selected objects, of a database.
ttSize	Estimates the amount of space that a given table, including any views in the database will consume when the table grows to include a specified number of rows.
ttStats	Monitors database metrics or takes and compares snapshots of metrics.
ttVersion	lists the TimesTen release information.

ttGridAdmin

Description

Use the `ttGridAdmin` utility for all aspects of administering a grid, such as creating a grid, adding or removing data instances or management instances, creating databases, and redistributing data to new data instances. The grid and database configuration resulting from these operations is stored in the grid model, which is distributed to instances of the grid. (See "Grid model" on page 4-4 for an introduction to the model.)

> **Important:** In TimesTen Scaleout, do not update configuration files manually (such as `timesten.conf`, `sys.odbc.ini`, and `tnsnames.ora`).

Required privilege

Instance administrator of the management instance from which `ttGridAdmin` is run. The user then becomes the instance administrator of all instances created with `ttGridAdmin`.

Usage with TimesTen Scaleout

This utility is specifically for use with TimesTen Scaleout, with commands that perform any operations on a grid.

Syntax

For general syntax (help options and options that apply to all commands), see "Help and general options" on page 4-10. For syntax for individual commands, see the relevant sections under "ttGridAdmin operations" on page 4-9.

Grid model

TimesTen Scaleout maintains a central configuration of the grid within a model that describes the desired structure of the grid. The model represents the desired logical topology of the grid and contains objects that represent components of the grid, such as installations, hosts, instances, and database definitions. Each time you use `ttGridAdmin` to add a grid component, a corresponding object is added to the model.

When you use `ttGridAdmin` to apply the model, TimesTen Scaleout attempts to implement it into the operational grid, such as by creating the desired physical installations and instances.

Grid objects and object naming

Each entity in a grid—such as each host, instance, installation, physical group, and repository—is defined as a named object.

Each object type, representing these types of entities, has its own namespace. You can have a host named xyz and a physical group named xyz without conflict. In addition, instance namespaces and installation namespaces are per host. You can have an instance named instance1 on host1 and an instance named instance1 on host2 without conflict.

Object-naming hierarchies such as this can be expressed in ttGridAdmin syntax using fully qualified names, *toplevelobject.nextlevelobect*. For example, host1.instance1 and host2.instance1. To specify an instance or installation in ttGridAdmin syntax, you need only specify *hostname* (instead of *hostname.instancename* or *hostname.installname*) if there is only one instance or installation (as applicable) on the host.

Operations on grid entities through ttGridAdmin, such as creating or removing an instance or installation, are managed through the corresponding objects in the model. The physical entities themselves are not created or removed until the model is applied through the ttGridAdmin modelApply command. For example, to remove an instance named host1.instance1, the instanceDelete command removes the object named host1.instance1 from the model, then the modelApply command removes the physical instance (the instance home directory and everything under it).

Be aware of these limitations in object naming in the grid model:

- Names must use only ASCII characters.

- Allowed characters are alphabetic, numeric, hyphen (-), and underscore (_).

- Database definition names and connectable names are limited to 32 characters (due to ODBC limitations).

- All other object names are limited to 256 characters.

- Object names are case-insensitive (so you cannot have an instance instance and an instance Instance on the same host), but are represented and shown as specified in ttGridAdmin commands. If you specify MyInstance in the instanceCreate command, that is how it will be shown.

- You cannot name anything All or Default, which are reserved names.

Address formats

Some `ttGridAdmin` commands, such as `gridCreate` and `hostCreate`, have options to specify the address or addresses used for internal and external communications. You can specify addresses as DNS names or IP addresses, although use of DNS names is more typical. IP addresses can be in either IPv6 or IPv4 format. For example:

- DNS name: `myhost.example.com`

- IPv6 address: `2001:b400:2000:834:26:3eff:fe07:5b83`

- IPv4 address: `192.0.2.1`

Database management operations

In a Grid environment, database management operations—`dbCreate`/`dbDestroy`, `dbLoad`/`dbUnload`, `dbOpen`/`dbClose`—are performed asynchronously by default. This is generally advisable, as such operations are not atomic and may take a long time. In a large grid, loading a database may succeed immediately on many hosts, take a little longer on others, and much longer on others. Some hosts may, in fact, be down when the operation was executed, so cannot perform the operation until they are back up.

By default, commands for these database operations return without waiting for completion, but they can optionally wait, with or without a timeout. With wait and a timeout, a command does not return until it has completed on all instances or reaches the timeout. With wait and no timeout, a command will never return if any instances are down. There are advantages and disadvantages to each approach, depending on factors such as how large the grid is. For a large grid, you may choose to proceed before the operation has completed on all instances, while on a small grid it may be more sensible to wait until it has completed on every instance.

TimesTen Scaleout tracks the state of a database, including each element of the database, and it is up to the user to check status of an operation (through the `dbStatus` command, optionally using the `-all` option for further details) to see how many instances have completed the operation. In particular, after loading the database, you can use the status information to determine if it has been loaded on enough instances for the database to be opened and users to start accessing it.

No command is provided to cancel an operation on any or all instances.

Command timeouts and waits

Note that as `ttGridAdmin` executes a command, it may run operating system commands as well. Using the top-level `-timeout` option, you can specify that `ttGridAdmin` will wait for the specified number of seconds for such operations to complete. If an operation does not complete within the specified number of seconds, the `ttGridAdmin` command being executed fails.

In addition, the `ttGridAdmin` database management commands `dbCreate`, `dbDestroy`, `dbload`, `dbUnload`, `dbOpen` and `dbClose` have a `-nowait`/`-wait` option. Each of these commands initiates a state change that is recorded in the active management instance of the grid. With a setting of `-nowait` (the default), the commands return immediately without waiting for the state change to complete. If `-wait` with a timeout value n is specified, `ttGridAdmin` will wait for up to n seconds for the state change to complete. If `-wait` is specified without a timeout, `ttGridAdmin` will wait without limit for the state change to complete.

ttGridAdmin operations

The listed sections provide detailed information about `ttGridAdmin` commands and functionality in these areas:

- Help and general options
- Return values
- Backup and restore operations
- Connectable operations
- Data space group operations
- Database definition operations
- Database operations
- Grid operations
- Host operations
- Import and export operations
- Installation operations
- Instance operations
- Management instance operations
- Membership operations
- Model operations
- Oracle Database operations
- Physical group operations
- Repository operations

Help and general options

These options work with any `ttGridAdmin` command or, for help, at the top level without any command.

```
ttGridAdmin [-h | -help | -?] [command]
```

```
ttGridAdmin [-o json] [-timeout n] command [-command_option] ...
```

Options

`ttGridAdmin` has the general options:

Option	Description
-h	Display help information.

-help
-?
	If specified by itself, this option categorizes and lists `ttGridAdmin` commands. For example:
	`ttGridAdmin -h`
	if specified with a command, this option displays syntax and option descriptions for the given command. For example:
	`ttGridAdmin -h dbCreate`

-o json	Choose JSON output. Output for the command will be in JSON format. (Otherwise, output is in human-readable format.) **Important**: There is no guarantee of JSON output compatibility between TimesTen releases.
-timeout n	Maximum number of seconds to wait for a long-running operation to complete. The default is 600. Note that as `ttGridAdmin` executes a command, it may run operating system commands as well. It will wait for the specified number of seconds for such operations to complete. If an operation does not complete within the specified number of seconds, the `ttGridAdmin` command being executed fails.

Return values

This section describes error or status values and JSON data elements returned by ttGridAdmin commands.

Error and status return codes

ttGridAdmin commands returns error codes to note success or failure, including these general codes:

Code	Description
0	Success
255	Internal error
254	Syntax error

JSON data elements returned

When JSON output is specified, the stdout of the command includes at least these name/value pairs. (Refer to http://www.json.org/ for general information about JSON output.)

Return	Type	Description
"status"	number	Return code
		See the preceding section, "Error and status return codes".
"errno"	number	Error number, if an error occurred
"errmsg"	string	Error message, if an error occurred

> **Note:** Additional, command-specific JSON data elements may also be returned.

Backup and restore operations

Use `ttGridAdmin` commands in this section to back up and restore databases, display the status of those operations, or delete a backup.

Also see "Migrating, Backing Up and Restoring Data" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Back up a database (dbBackup)

The `dbBackup` command initiates a backup of the specified database.

```
ttGridAdmin dbBackup dbname
                    -repository reponame
                    [-name backupname]
                    [-backupType normal|staged]
                    [-bwlimit limit]
                    [-compress value]
```

In some cases you must use `dbExport` instead. This would be the case, for example, if the grid topology at the restore location has fewer replica sets than the backed up database, or the restore location is running a version of TimesTen that is not patch-compatible with the version of the backed up database. See "Migrating, Backing Up and Restoring Data" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for additional information.

TimesTen Scaleout enables you to create staged backups for SCP repositories. This type of backup eliminates the overhead of creating local copies of the checkpoints and log files and reduces the WAN traffic of creating a remote copy in the repository. See "Back up a database into a remote repository (WAN-friendly)" in the *Oracle TimesTen In-Memory Database Scaleout User's Guide* for more information.

> **Important:** Be aware of the following if the specified repository was created with `-method scp`:
>
> For normal backups, backup file for each element is stored on the local file system where the element is located before being copied to the remote repository.

A backup is stored as a *collection* under a *repository*. You first must create the repository. See "Repository operations" on page 4-110.

Options

The `dbBackup` command has the options:

Option	Description
dbname	Name of the database to back up.
-repository *reponame*	Name of the repository where the backup will be located.
-name *backupname*	Specifies a name for the backup. The default is the letter "B" followed by the date and time of the backup, in the format: B*yyyymmddhhmmss*

Option	Description
-backupType normal|staged	For repositories using the SCP method, specifies the type of backup to create. Supported options are normal or staged.
	■ normal: The checkpoint and log files of one element for each replica set are temporarily copied to the instance home of such elements before being copied to the repository.
	■ staged: The checkpoint and log files of one element for each replica set are temporarily used as target for symbolic links in the instance home of such elements before being synchronized with a staging directory on the repository. The resulting files are then copied to the backup location on the repository.
	By default, TimesTen creates normal backups.
-bwlimit limit	For staged backups, specifies the aggregated maximum bandwidth (in MB per second) used to copy and synchronize files between hosts and repository.
	By default, staged backups use as much WAN bandwidth as possible.
-compress value	For staged backups, specifies the level of compression used to copy and synchronize files between hosts and repository. Supported values range from 0 to 9, where 0 represents no compression and 9 represents the maximum compression available.
	By default, staged backups use no compression.

Examples

This example backs up database1 into repository repo1. It uses the default name for the backup, according to the current timestamp (2/22/17 at 14:55:44).

```
% ttGridAdmin dbBackup database1 -repository repo1
dbBackup B20170222145544 started
```

You can then use dbBackupStatus to check progress, as shown in the example in "Display the status of a database backup (dbBackupStatus)" on page 4-14. The backup is finished when each element and the database as a whole are indicated as complete.

Notes

- The backup is performed asynchronously. Use the dbBackupStatus command to check progress.

- One element from each replica set is backed up.

- Each replica set is stored as a sub-collection.

- For disk space requirements, see "Backing up and restoring a database" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Delete a database backup (dbBackupDelete)

The dbBackupDelete command deletes the specified database backup.

```
ttGridAdmin dbBackupDelete -repository reponame
                            -name backupname
```

Options

The `dbBackupDelete` command has the options.

Option	Description
`-repository` *reponame*	Name of the repository where the backup is located.
`-name` *backupname*	Name of the backup to delete.

Examples

This example deletes the backup created in the example in "Back up a database (dbBackup)" on page 4-12.

```
% ttGridAdmin dbBackupDelete -repository repo1 -name B20170222145544
Backup B20170222145544 deleted
```

Notes

This command is typically used to delete old or failed backups.

Display the status of a database backup (dbBackupStatus)

The `dbBackupStatus` command shows the status of a database backup or backups previously started.

```
ttGridAdmin dbBackupStatus dbname
                          [-name backupname]
```

Options

The `dbBackupStatus` command has the options:

Option	Description
dbname	Name of the database being backed up.
`-name` *backupname*	Name of the backup to check. The default is all backups of the specified database.

Examples

This example shows status upon completion of the backup from the example in "Back up a database (dbBackup)" on page 4-12.

```
% ttGridAdmin dbBackupStatus database1 -name B2017022245544
Database  Backup          Repository Host  Instance  Elem State     Started               Finished
--------- --------------- --------- ----- --------- ---- --------- --------------------- --------
database1 B20170222145544 repo1                          Completed 2017-02-22T14:55:44.000Z Y
                                    host3 instance1    1 Complete
                                    host4 instance1    2 Complete
                                    host5 instance1    3 Complete
```

Notes

- When you believe the backup is complete, confirm that `dbBackupStatus` shows that the backup as a whole and for each instance is shown as complete. If there were any failures, see "Check the status of a backup" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

- `Y` in the `Finished` column indicates that the command has finished executing, regardless of state—that each instance has succeeded or failed.

- The metadata associated with a database backup is deleted when the database is deleted. After a database is deleted, you can use the `repositoryList` command to see existing backups.

Restore a database (dbRestore)

The `dbRestore` command restores a database backup into a new database.

```
ttGridAdmin dbRestore dbname
                      -repository reponame
                      -backup backupname
```

Options

The `dbRestore` command has the options:

Option	Description
dbname	Name of the database to be created, then restored from the backup.
`-repository` *reponame*	Name of the repository where the backup is located.
`-backup` *backupname*	Name of the backup to use for the restore.

Examples

This example creates and restores a database `res_db1` from a backup `mybkup`.

```
% ttGridAdmin dbRestore res_db1 -repository repo1 -backup mybkup
dbRestore mybkup started
```

You can then use `dbRestoreStatus` to check progress, as shown in the example in The restore is finished when each element and the database as a whole are indicated as complete.

Notes

- This database must already be defined (with `dbdefCreate`) but not yet created.

- The restore is performed asynchronously. Use the `dbRestoreStatus` command to check progress.

- The restored database is loaded into memory when `dbRestore` completes, but not opened.

- You can restore to the original database definition or to a newly created database definition.

- You cannot restore to a database with fewer replica sets than what was backed up. (If the number of data instances on hosts in each data space group is not sufficient to support the number of replica sets in the database that was backed up, you must use `dbExport` and `dbImport` instead.)

- If you restore to a database with more replica sets than what was backed up, only the number of replica sets that were backed up will be added to the database distribution map. For example, if you back up a database with two replica sets and restore to a database with four replica sets, only the elements in two replica sets will be added to the distribution map. You would then have to redistribute data with `dbDistribute` to get four replica sets.

■ For disk space requirements, see "Backing up and restoring a database" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Display the status of a database restore (dbRestoreStatus)

The dbRestoreStatus command shows the status of a database restore previously started.

```
ttGridAdmin dbRestoreStatus dbname
```

Options

The dbRestoreStatus command has the option:

Option	Description
dbname	Name of the database where the restore is being checked.

Examples

This example shows status upon completion of the restore from the example in "Restore a database (dbRestore)" on page 4-15.

```
% ttGridAdmin dbRestoreStatus res_db
Database Restore Repository Host  Instance  Elem State                        Started               Finished
-------- ------- ---------- ----- --------- ---- -------------------------------------------------- -
res_db1  mybkup  repo1                            Restore_Finale_Complete 2017-03-03T13:19:39.000Z Y
                            host3 instance1       Restore_Instance_Complete
                            host4 instance1       Restore_Instance_Complete
                            host5 instance1       Restore_Finale_Complete
```

Notes

■ When you believe the restore is complete, confirm that dbRestoreStatus shows Restore_Finale_Complete for the restore as a whole and Resore_Instance_ Complete or Restore_Finale_Complete for each instance. If there were any failures, see "Check the status of a restore" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

■ Y in the Finished column indicates that the command has finished executing, regardless of state—that each instance has succeeded or failed.

Connectable operations

Use `ttGridAdmin` commands in this section to create, delete, modify, export, or list connectable objects, used in connecting to a TimesTen Scaleout database. A connectable specifies a set of connection attribute settings and thereby defines an underlying DSN and `tnsnames.ora` file entry.

There are two types of connectables: direct connectables for direct mode access, and client/server connectables for client/server access.

You can have multiple connectables for a database to use different sets of connection attribute settings. For example, if you have one application designed to receive ASCII data from the database and another designed to receive Unicode data, then create two connectables, each with the appropriate `ConnectionCharacterSet` attribute setting.

> **Note:** A direct connectable, with the same name as the database, is created automatically when you define a database with the `dbdefCreate` command.

See "Connecting to a database" and its subsections in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for additional information about connectables.

Create a connectable (connectableCreate)

The `connectableCreate` command creates a connectable object in the model, defining connection attribute settings.

```
ttGridAdmin connectableCreate -dbdef name
                                   [-cs [-only hostname[.instancename]]]
                                   filepath
```

Options

The `connectableCreate` command has the options:

Option	Description
-dbdef *name*	The name of the database definition object that the connectable references (which is the name of the database to connect to).
-cs	Specifies that the connectable will be for client/server access to the database. If this is not specified, the connectable will be for direct mode access.
-only *hostname*[.*instancename*]	For client/server connections, optionally specifies that applications connecting through this connectable should connect to the indicated instance. By default, all data instances in the grid are available and connections are distributed among them.
	You need to specify the instance name only if there is more than one instance on the host. (See "Grid objects and object naming" on page 4-5.)
	You can specify multiple -only options, in which case client/server applications can connect to any instance listed in the -only settings.

Option	Description
filepath	Path and name of the connectable file, which contains the connection attribute settings for the connectable. The file name must be of the form *connname*.connect, where *connname* defines the name of the connectable.
	Also see the example below.

Examples

The example creates a client/server connectable using this connectable file, database1client.connect:

```
ConnectionCharacterSet=AL32UTF8
UID=ttclient
```

Create the connectable:

```
% ttGridAdmin connectableCreate -dbdef database1 -cs
/sw/tten/grid/conndefs/database1client.connect
Connectable database1client created.
```

Notes

- The connectable file must be in odbc.ini format, as shown in the example, with *attribute*=*value* on each line. A DSN name, in [*name*] format such as [database1client] here, is optional. If provided, it must match the connectable name determined by the connectable file name.

- The default value is used for any connection attribute not set in the connectable.

- When you apply the model after creating a connectable, new versions of all necessary configuration files are written to each data instance. This makes the connectable available for use. (Never edit configuration files manually. They are overwritten each time the model is applied.)

- You cannot set data store (creation time) attributes or first connection attributes in the connectable file. Those must be set in the database definition file for the dbdefCreate or dbdefModify command.

- Any settings for TTC_SERVER, TTC_SERVER_DSN, or TCP_PORT are ignored. Appropriate values for the grid topology are automatically used.

- The connectable that is defined is usable through all APIs supported by TimesTen.

Delete a connectable (connectableDelete)

The connectableDelete command deletes an existing connectable object from the model.

```
ttGridAdmin connectableDelete name
```

Options

The connectableDelete command has the option:

Option	Description
name	The name of the connectable object to delete.

Examples

```
% ttGridAdmin connectableDelete database1client
Connectable database1client deleted from Model.
```

Notes

When you apply the model after deleting a connectable, new versions of all necessary configuration files are written to each data instance, with the connectable entry removed. (Never edit configuration files manually. They are overwritten each time the model is applied.)

Export a connectable (connectableExport)

The `connectableExport` command exports a connectable object and its connection attribute settings, typically to a specified file.

```
ttGridAdmin connectableExport name
                         [filepath]
```

Options

The `connectableExport` command has the options:

Option	Description
name	Name of the connectable object to export.
filepath	Path and name of the connectable file to create, typically a `.connect` file for use with `connectableCreate` or `connectableModify`. If no file is specified, the export goes to `stdout`.
	Important: If you specify an existing file, it will be overwritten.

Examples

This example exports the connectable created in the `connectableCreate` example to the file database1client.connect.

```
% ttGridAdmin connectableExport database1client
/sw/tten/grid/conndefs/database1client.connect
Connectable database1client exported
```

Resulting contents of database1client.connect:

```
# Connectable GUID 3210288C-DF44-447D-ADB6-BDC8F7CFE17C Exported 2017-11-14
17:53:25
[database1client]
ConnectionCharacterSet=AL32UTF8
UID=ttclient
```

Notes

A typical use case is if you want to modify a connectable, but the original connectable file is no longer available.

List connectables (connectableList)

The `connectableList` command lists the connectable objects that have been created in the specified version of the model.

```
ttGridAdmin connectableList [-latest|-current|-version n]
                         [-dbdef]
```

Options

The `connectableList` command has the options:

Option	Description
-latest	Lists connectable objects in the latest model—the model being modified and not yet applied to the grid. This is the default.
-current	Lists connectable objects in the current model—the model most recently applied to the grid.
-version *n*	Lists connectable objects in the specified version number of the model.
-dbdef	Also lists the name of the database definition associated with each connectable object.

Examples

This example lists connectables and the associated database definitions from the model most recently applied to the grid.

```
% ttGridAdmin connectableList -current -dbdef
Connectable     DbDef
--------------- -----------
database1       database1
database1client database1
```

Only database1client was created by the user. A direct connectable is also automatically defined for each user-created database definition.

Modify a connectable (connectableModify)

The `connectableModify` command modifies a connectable object.

```
ttGridAdmin connectableModify [-only hostname[.instancename]]
                              filepath
```

Options

The `connectableModify` command has the options:

Option	Description
-only *hostname*[.*instancename*]	For connectables created for client/server connections (as determined by the -cs option of connectableCreate), optionally specifies that applications connecting through this connectable should connect to the indicated instance. By default, all data instances in the grid are available and connections are distributed among them.
	You need to specify the instance name only if there is more than one instance on the host.
	You can specify multiple -only options, in which case client/server applications can connect to any instance listed in the -only settings.
	Note: Instances specified with connectableModify -only replace any instances specified with connectableCreate -only.
filepath	Path and name of the connectable file, which contains the new set of connection attribute settings. The file name must be of the form *connname*.connect, where *connname* is the name of the connectable.

Examples

The example modifies the client/server connectable created in the `connectableCreate` example, editing database1client.`connect` to add a `PermWarnThreshold` setting:

```
ConnectionCharacterSet=AL32UTF8
UID=ttclient
PermWarnThreshold=80
```

Modify the connectable:

```
% ttGridAdminttGridAdmin connectableModify
/sw/tten/grid/conndefs/database1client.connect
Connectable database1client modified.
```

Notes

- All connection attribute settings from the previous connectable file are replaced with the connection attribute settings in the specified file.

- You cannot modify a connectable to change from client/server use to direct mode use, or from direct mode use to client/server use. Instead, delete the connectable and create a new one.

- When you apply the model after modifying a connectable, new versions of all necessary configuration files are written to each data instance, with the connectable entry modified according to the new settings. (Never edit configuration files manually. They are overwritten each time the model is applied.)

Data space group operations

Use the `ttGridAdmin` command in this section to get recommendations for assignments of hosts to data space groups.

Get recommendations for data space group assignments (dataSpaceGroupSuggest)

The `dataSpaceGroupSuggest` command recommends data space group assignments for hosts, based on the physical topology of the grid as defined in the model. Recommendations in the form of appropriate `hostModify` commands are output to a shell script (or to `stdout`).

```
ttGridAdmin dataSpaceGroupSuggest [filepath]
```

Options

The `dataSpaceGroupSuggest` command has the option:

Option	Description
filepath	Path and name of the script to contain the recommended actions.
	If no file is specified, the script is written to `stdout`.

Examples

This example shows a `ttGridAdmin dataSpaceGroupSuggest` command that writes its recommendations to `recommendations.sh`. (The output is abbreviated.) If you then execute this file, it will run `ttGridAdmin hostModify` commands to assign hosts to the recommended data space groups.

```
% ttGridAdmin dataSpaceGroupSuggest /tmp/recommendations.sh
% more /tmp/recommendations.sh
#!/bin/sh
# Recommendations generated by ttGridAdmin -dataSpaceGroupSuggest at Wed Aug  8
17:40:35 PDT 2018

TIMESTEN_HOME=/sw/tten/grid/ttinstances/grid1_mgmt
export TIMESTEN_HOME
. $TIMESTEN_HOME/bin/ttenv.sh > /dev/null 2>/dev/null

# Number of possibilities evaluated:     126
#
# Number of usable possibilities found:  10
# (A 'usable' possibility is one that is compatible with pre-existing
# assignments of Hosts to DataSpaceGroups)
#
# Number of 'ideal' possibilities found: 1
# (An 'ideal' possibility is one where no PhysicalGroups span multiple
# DataSpaceGroups)
#
# Possibilities evaluated (best 10 displayed):
# ...
#
# This script, if executed, would implement the only 'ideal' configuration found.
# Even though this recommendation was 'ideal', you should carefully evaluate it
# prior to running this script.
# Host mysys1host is already in DataSpaceGroup 1
ttGridAdmin hostModify mysys7host -dataSpaceGroup 1
ttGridAdmin hostModify mysys8host -dataSpaceGroup 1
```

```
# Host mysys3host is already in DataSpaceGroup 1
# Host mysys4host is already in DataSpaceGroup 1
# Host mysys5host is already in DataSpaceGroup 2
ttGridAdmin hostModify mysys9host -dataSpaceGroup 2
ttGridAdmin hostModify mysys10host -dataSpaceGroup 2
ttGridAdmin hostModify mysys2host -dataSpaceGroup 2
# Host mysys6host is already in DataSpaceGroup 2
```

If you decide to accept these recommendations, execute the provided shell script, `recommendations.sh`. Once the script is executed, all hosts are assigned to the designated data space groups.

```
% sh /scratch/gridsetup/misc/recommendations.sh
Host mysys7host modified in Model
Host mysys8host modified in Model
Host mysys9host modified in Model
Host mysys10host modified in Model
Host mysys2host modified in Model
```

Notes

- Run the command after you have created physical groups, hosts, and installations for the grid and defined how hosts will correspond to physical groups (which you can accomplish through the `hostCreate -physicalgroup` option or the `hostModify -physicalgroup` or `-addphysicalgroup` option).

- This command is for hosts not yet assigned to a data space group, and returns an error if all hosts are already assigned. (Once a host is assigned to a data space group and `modelApply` is executed, you cannot change the assignment.)

- Hosts that do not have any instances are not included in the output.

- Recommendations are based on separating hosts among data space groups in an effort to avoid single points of failure, according to shared physical resources. In addition to the `hostModify` commands, the script has comments explaining the rationale behind the recommendations.

- If there is not enough difference in physical locations for `dataSpaceGroupSuggest` to make meaningful suggestions, it will output a message to that effect:

  ```
  Error 33: Not enough differentiation to make suggestions
  ```

- For additional information, see "Assigning hosts to data space groups" and "Propose data space group assignments" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Database definition operations

Use `ttGridAdmin` commands in this section to create, delete, export, or modify a database definition, or to display a list of existing database definitions.

A database definition specifies the characteristics of a database, consisting of settings for data store attributes (set at database creation) and first connect attributes.

Create a database definition (dbdefCreate)

The `dbdefCreate` command creates a database definition object in the model, defining characteristics of the database according to attribute settings in the specified file. It also creates a direct connectable object with the same name as the database, for direct connections.

```
ttGridAdmin dbdefCreate filepath
```

Once a database definition is added to the model, it can be used to create a database.

Options

The `dbdefCreate` command has the option:

Option	Description
filepath	Path and name of the database definition file, which contains the attribute settings for the database definition.
	The file name must be of the form *dbname*`.dbdef`, where *dbmame* defines the name of the database.

Examples

The example uses database definition file `database1`.`dbdef`:

```
DataStore=/disk/databases/database1
LogDir=/disk2/logs
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=US7ASCII
PermSize=256
TempSize=128
```

Typical settings in a database definition file include the following, as shown in the example.

- Data store attributes: `DataStore` (required) `LogDir`, and `DatabaseCharacterSet` (required)

  Directories are created on each host as necessary for the `DataStore` and `LogDir` locations.

- First connection attributes: `PermSize` (required) and `TempSize`

- General connection attribute: `ConnectionCharacterSet`

- As necessary, PL/SQL first connection attributes and server connection attributes

Create the database definition object:

```
% ttGridAdmin dbdefCreate  /sw/tten/grid/dbdefs/database1.dbdef
Database Definition database1 created.
```

Data store attributes and first connection attributes go in the resulting database definition. From the example, this consists of DataStore, LogDir, DatabaseCharacterSet, PermSize, and TempSize. In addition, a default connections setting and a default durability setting are added to the definition automatically.

```
[database1]
DataStore=/disk/databases/database1
DatabaseCharacterSet=AL32UTF8
LogDir=/disk2/logs
PermSize=256
TempSize=128
connections=100
durability=0
```

General connection attributes go in the resulting connectable definition that is automatically created. In the example, this consists of ConnectionCharacterSet:

```
[database1]
ConnectionCharacterSet=US7ASCII
```

For additional information, see "Creating a database definition file" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Notes

- The database name cannot be the same as any existing database or connectable name.

- The database definition file must be in odbc.ini format, as shown in the example, with *attribute=value* on each line.

- A dbdef file supports the following substitution strings for DataStore and LogDir entries. They are replaced as appropriate when the model is applied:

  - !!HOST_NAME!! is replaced on each host by the host name as specified in the model.

  - !!INSTANCE_NAME!! is replaced in each instance by the instance name as specified in the model.

  A scenario for using !!HOST_NAME!! and !!INSTANCE_NAME!!, for example, would be if you use a Storage Area Network device shared between the hosts of the grid. Setting DataStore to /shared/datastores/!!HOST_NAME!!/!!INSTANCE_NAME!! gives each host its own data storage area.

- It is best practice to specify LogDir and have it be on a different file system from DataStore. The DataStore and LogDir paths and directories will be created wherever necessary.

- You can create additional connectables as described in "Create a connectable (connectableCreate)" on page 4-17.

- Some connection attributes cannot be set in a dbdef file, although they could be set for additional connectables that you create. For example, because the initial connectable that is created during database definition must be usable by the instance administrator, the UID and PWD connection attributes cannot be specified in a dbdef file.

- When you apply the model after creating a database definition, new versions of all necessary configuration files are written to each data instance, with an entry added according to the dbdef settings. (Never edit configuration files manually. They are overwritten each time the model is applied.)

Delete a database definition (dbdefDelete)

The `dbdefDelete` command removes a database definition object from the model.

```
ttGridAdmin dbdefDelete name
                        [-cascade|-nocascade]
```

Options

The `dbdefDelete` command has the options:

Option	Description
name	The name of the database definition object to delete.
-cascade	Also remove any additional connectable objects that were created for this database. This is the default.
-nocascade	Do not remove connectable objects.

Examples

This example deletes the database definition object `database2`, showing database definition object listings before and after.

```
% ttGridAdmin dbdefList
database1
database2

% ttGridAdmin dbdefDelete database2
Database Definition database2 deleted.

% ttGridAdmin dbdefList
database1
```

Notes

- The database itself must have already been destroyed (or not yet been created).

- When you apply the model after deleting a database definition, new versions of all necessary configuration files are written to each data instance, with the database definition removed. (Never edit configuration files manually. They are overwritten each time the model is applied.)

- The connectable object that was automatically created when the database definition object was created is also removed, regardless of the -cascade/-nocascade setting.

Export a database definition (dbdefExport)

The `dbdefExport` command exports an existing database definition object from the model, typically to a specified file.

```
ttGridAdmin dbdefExport name
                        [filepath]
```

Options

The `dbdefExport` command has the options:

Option	Description
name	Name of the database definition to export.
filepath	Path and name of the database definition file to create, typically a .dbdef file for use with dbdefCreate or dbdefModify. If no file is specified, the export goes to stdout. **Important**: If you specify an existing file, it will be overwritten.

Examples

This example exports the database definition created in the dbdefCreate example above to the file database1.dbdef.

```
% ttGridAdmin dbdefExport database1 /sw/tten/grid/dbdefs/database1.dbdef
```

Resulting contents of database1.dbdef:

```
# DbDef GUID BCC6AB97-FDC2-4453-AEBC-5BFCAA57EA52 Exported 2017-12-06 19:05:03
[database1]
DataStore=/disk/databases/database1
DatabaseCharacterSet=AL32UTF8
LogDir=/disk2/logs
PermSize=256
TempSize=128
connections=100
durability=0
```

Notes

- The database definition is exported in odbc.ini format, as shown in the example, with *attribute*=*value* on each line.

- A typical use case is if you want to modify a database definition, but the original database definition file is no longer available.

List database definitions (dbdefList)

The dbdefList command lists the database definition objects that exist in the specified version of the model.

```
ttGridAdmin dbdefList [-latest|-current|-version n]
```

Options

The dbdefList command has the options:

Option	Description
-latest	Lists database definition objects in the latest model—the model being modified and not yet applied to the grid. This is the default.
-current	Lists database definition objects in the current model—the model most recently applied to the grid.
-version *n*	Lists database definition objects in the specified version number of the model.

Examples

List database definition objects in the latest model (default) after the database1 database definition object was created (as shown in "Create a database definition

(dbdefCreate)" on page 4-24).

```
% ttGridAdmin dbdefList
database1
```

Modify a database definition (dbdefModify)

The `dbdefModify` command modifies an existing database definition object in the model, defining characteristics of the database according to attribute settings in the specified file.

```
ttGridAdmin dbdefModify filepath
```

Options

The `dbdefModify` command has the option:

Option	Description
filepath	Path and name of the file containing the database definition that will modify the database definition object.
	The file name must be of the form *dbname*.dbdef, where *dbname* is the name of the database.

Examples

This example modifies database1, created in "Create a database definition (dbdefCreate)" on page 4-24. The database definition file database1.dbdef was updated to change the `PermSize` and `TempSize`:

```
% cd /sw/tten/grid/dbdefs
% more database1.dbdef
DataStore=/disk/databases/database1
LogDir=/disk2/logs
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=US7ASCII
PermSize=512
TempSize=256
```

Modify the database definition object:

```
% ttGridAdmin dbdefModify  /sw/tten/grid/dbdefs/database1.dbdef
Database Definition database1 modified.
```

Notes

- Database definition files are in `odbc.ini` format, as shown in the example, with *attribute*=*value* on each line. See "Create a database definition (dbdefCreate)" on page 4-24 for additional discussion.

- Data store attributes, as listed in Table 2–1 (for example, `DataStore`, `DatabaseCharacterSet`, `LogDir`, and `Durability`), are frozen once a database is created. Trying to change them using `dbdefModify` will have no effect on the database.

- If the database exists and is loaded, changes by the `dbdefModify` command to first connection attributes do not take effect until you unload (`dbUnload`) and load (`dbLoad`) the database.

- This command does not modify the database itself, only the database definition object.

- The connectable object that was automatically created when the database definition object was created is also modified appropriately.

- When you apply the model after modifying a database definition, new versions of all necessary configuration files are written to each data instance, with the applicable entry modified according to the dbdef settings. (Never edit configuration files manually. They are overwritten each time the model is applied.)

- The specified definition completely replaces the previous definition, deleting previous attribute settings for the database definition and connectable definition. Attributes that were set previously but are not set in the new definition will take their default settings.

Database operations

Use `ttGridAdmin` commands in this section to perform operations on databases, including creating, destroying, loading, unloading, opening, closing, importing, and exporting a database; setting the distribution scheme of a database; determining status of a database; and forcing the termination of connections to the database.

Close a database (dbClose)

The `dbClose` command closes the database so that applications can no longer connect to it.

```
ttGridAdmin dbClose name
                    [-nowait | -wait [timeout]]
```

Options

The `dbClose` command has the options:

Option	Description
name	Name of the database to close.
-nowait | -wait [*timeout*]	The command initiates a state change that is recorded in the active management instance of the grid.
	The -nowait option causes the command to return immediately without waiting for the state change. This is the default behavior.
	The -wait option causes the command to wait for the state change to complete, when the database element has been closed on each instance in the grid. You can optionally subject the wait to a limit of *timeout* seconds. Otherwise, or if *timeout* is set to 0, there is no limit.
	In a large grid, it is not typical or generally advisable to use -wait. If you do, it is advisable to set a timeout. (See "Database management operations" on page 4-7.)

Examples

This example closes a database without waiting for the elements to be closed on all instances, then checks status (after the database was successfully closed):

```
% ttGridAdmin dbClose database1
Database database1 close started
...
% ttGridAdmin dbStatus database1
Database database1 summary status as of Mon Nov 13 19:27:48 PST 2017

created,loaded-complete,closed
Completely created elements: 4 (of 4)
Completely loaded elements: 4 (of 4)
Completely created replica sets: 2 (of 2)
Completely loaded replica sets: 2 (of 2)

Open elements: 0 (of 4)
```

Notes

- After the command has completed, the database is still loaded but is closed to connections. Only the instance administrator can connect to a closed database.

- If you run dbClose asynchronously (without waiting), you can use the dbStatus command to see when the database is closed.

- The command does not close existing database connections. Any previously open connections must be terminated independently.

- If a database has been closed with dbClose, attempting to close it again typically results in an error. However, if any element is in "close failed" state, you can retry dbClose. Doing so will change any element in "close failed" state to "opened" state, which will result in TimesTen Scaleout trying to close it again.

Create a database (dbCreate)

The dbCreate command creates a database in the grid according to the specified database definition.

```
ttGridAdmin dbCreate name
                    [-instance hostname[.instancename]]
                    [-nowait | -wait [timeout]]
```

Options

The dbCreate command has the options:

Option	Description
name	Name of the database definition to use in creating the database. This becomes the name of the database.
-instance hostname[.instancename]	If specified, database element(s) will be created only on the specified instance(s), instead of on all instances of the grid. Any element previously created successfully on any of the specified instances must first be destroyed.
	This is typically used to recover after a failure in the grid or after database elements were not successfully created on one or more instances in a previous execution of dbCreate.
	The hostname is required. The instancename is required only if there is more than one instance on the host. (See "Grid objects and object naming" on page 4-5.)
	You can use this option only once, specifying a single instance, in a single command.

Option	Description
-nowait | -wait [*timeout*]	The command initiates a state change that is recorded in the active management instance of the grid.
	The -nowait option causes the command to return immediately without waiting for the state change. This is the default behavior.
	The -wait option causes the command to wait for the state change to complete, when the database element has been created on each instance in the grid. You can optionally subject the wait to a limit of *timeout* seconds. Otherwise, or if *timeout* is set to 0, there is no limit.
	In a large grid, it is not typical or generally advisable to use -wait. If you do, it is advisable to set a timeout. (See "Database management operations" on page 4-7.)

Examples

This example creates a database without waiting for the elements to be created on all instances, then checks the status, first while database creation is still in progress, then after it is complete.

```
% ttGridAdmin dbCreate database1
Database database1 creation started
...
% ttGridAdmin dbStatus database1
Database database1 summary status as of Mon Nov 13 18:38:39 PST 2017

creating,loading-partial,closed
Completely created elements: 1 (of 4) (3 in progress)
Completely loaded elements: 1 (of 4) (3 in progress)
Completely created replica sets: 0 (of 0)
Completely loaded replica sets: 0 (of 0)

Open elements: 0 (of 4)
...
% ttGridAdmin dbStatus database1
Database database1 summary status as of Mon Nov 13 18:39:16 PST 2017

created,loaded,closed
Completely created elements: 4 (of 4)
Completely loaded elements: 4 (of 4)
Completely created replica sets: 0 (of 0)
Completely loaded replica sets: 0 (of 0)

Open elements: 0 (of 4)
```

In the following example, element creation on one instance fails. The example tries again to create the element on that instance after the problem is resolved.

```
% ttGridAdmin dbCreate database1
Database database1 creation started
...
% ttGridAdmin dbStatus database1 -all
Database database1 summary status as of Sat Nov 11 14:23:05 PST 2017

created-partial,loaded,closed
Completely created elements: 3 (of 4)(1 failed)
Completely loaded elements: 3 (of 4)
```

```
Completely created replica sets: 0 (of 0)
Completely loaded replica sets: 0 (of 0)

Open elements: 0 (of 4)

Database database1 element level status as of Sat Nov 11 14:23:05 PST 2017

Host        Instance  Elem Status Date/Time of Event  Message
---------- --------- ---- ------ ------------------ -------
mysys3host griddata1    1 loaded 2017-11-11 14:22:52
mysys4host griddata2    2 loaded 2017-11-11 14:22:51
mysys5host griddata3    3 failed 2017-11-11 14:22:52
mysys6host griddata4    4 loaded 2017-11-11 14:22:53

Database database1 Replica Set status as of Sat Nov 11 14:23:05 PST 2017

RS DS Elem Host Instance Status Date/Time of Event Message
-- -- ---- ---- -------- ------ ------------------ -------

Database database1 Data Space Group status as of Sat Nov 11 14:23:05 PST 2017

DS RS Elem Host Instance Status Date/Time of Event Message
-- -- ---- ---- -------- ------ ------------------ -------
```

(Resolve the problem with mysys5host.griddata3.)

```
% ttGridAdmin dbCreate database1 -instance mysys5host.griddata3
Database database1 creation started
...
% ttGridAdmin dbStatus database1
Database database1 summary status as of Mon Nov 13 13:44:12 PST 2017

created,loaded,closed
Completely created elements: 4 (of 4)
Completely loaded elements: 4 (of 4)
Completely created replica sets: 0 (of 0)
Completely loaded replica sets: 0 (of 0)

Open elements: 0 (of 4)
```

Notes

- Each instance creates its element of the database, loads the element into memory, and records the state of the element.

- If you run dbCreate asynchronously (without waiting), you can use the dbStatus command to see when the database is created.

- The database is marked as "existing" as soon as the dbCreate command returns. If you run the command in the default -nowait mode, you can unload the database while its creation is still in progress.

- The database is not available for connections from users other than the instance administrator until you define the database distribution map with dbDistribute and open the database with dbOpen.

- A typical use case for the -instance option is when an element of the database had previously failed, been evicted or removed from the database distribution map, and been destroyed. (Then also use dbDistribute to add the element to the distribution map.)

Destroy a database (dbDestroy)

The dbDestroy command destroys the specified database. All data and schema contained in the database are irretrievably lost.

```
ttGridAdmin dbDestroy name
                    [-instance hostname[.instancename]]
                    [-nowait | -wait [timeout]]
```

Options

The dbDestroy command has the options:

Option	Description
name	Name of the database to destroy.
-instance hostname[.instancename]	If specified, database element(s) will be destroyed only on the specified instance(s), but elements will remain on all other instances of the grid. The elements to destroy must have been previously evicted or removed from the database distribution map or never added to the distribution map.
	The hostname is required. The instancename is required only if there is more than one instance on the host.
	You can use this option only once, specifying a single instance, in a single command.
-nowait	-wait [timeout]	The command initiates a state change that is recorded in the active management instance of the grid.
	The -nowait option causes the command to return immediately without waiting for the state change. This is the default behavior.
	The -wait option causes the command to wait for the state change to complete, when the database element has been destroyed on each instance in the grid. You can optionally subject the wait to a limit of timeout seconds. Otherwise, or if timeout is set to 0, there is no limit.
	In a large grid, it is not typical or generally advisable to use -wait. If you do, it is advisable to set a timeout. (See "Database management operations" on page 4-7.)

Examples

This example destroys a database without waiting for the elements to be destroyed on all instances. A subsequent attempt to check status indicates that the database was successfully destroyed.

```
% ttGridAdmin dbDestroy database1
Database database1 destroy started
...
% ttGridAdmin dbStatus database1
Error 2: Database database1 does not exist
```

This example destroys two of the four elements in the database. Both elements are from the same replica set and had previously been evicted.

```
% ttGridAdmin dbDestroy database1 -instance mysys3host.griddata1
Database database1 instance mysys3host.griddata1 destroy started
% ttGridAdmin dbDestroy database1 -instance mysys4host.griddata2
```

```
Database database1 instance mysys4host.griddata2 destroy started
% ttGridAdmin dbStatus database1 -all
Database database1 summary status as of Tue Jan  9 16:04:16 PST 2018

created,unloaded,closed
Completely created elements: 2 (of 4)
Completely loaded elements: 0 (of 4)
Completely created replica sets: 1 (of 1)
Completely loaded replica sets: 0 (of 1)

Open elements: 0 (of 2)

Database database1 element level status as of Tue Jan  9 16:04:16 PST 2018

Host        Instance  Elem Status    Date/Time of Event  Message
----------  --------- ---- --------- ------------------- -------
mysys3host  griddata1    1 destroyed 2018-01-09 16:04:02
mysys4host  griddata2    2 destroyed 2018-01-09 16:04:01
mysys5host  griddata3    3 unloaded  2018-01-09 16:01:25
mysys6host  griddata4    4 unloaded  2018-01-09 16:01:01

Database database1 Replica Set status as of Tue Jan  9 16:04:16 PST 2018

RS DS Elem Host        Instance  Status   Date/Time of Event  Message
-- -- ---- ----------  --------- -------- ------------------- -------
 1  1    3 mysys5host  griddata3 unloaded 2018-01-09 16:01:25
 1  2    4 mysys6host  griddata4 unloaded 2018-01-09 16:01:01

Database database1 Data Space Group status as of Tue Jan  9 16:04:16 PST 2018

DS RS Elem Host        Instance  Status Date/Time of Event  Message
-- -- ---- ----------  --------- ------ ------------------- -------
 1  1    3 mysys5host  griddata3 unloaded 2018-01-09 16:01:25
 2  1    4 mysys6host  griddata4 unloaded 2018-01-09 16:01:01
```

Notes

- The database must be unloaded or unloading.

- If you run dbDestroy asynchronously (without waiting), you can use the dbStatus command to see when the database is removed.

- A typical use case for the -instance option is after an element of the database failed and was evicted or removed from the database distribution map. Then using dbDestroy with -instance recovers the disk space of the failed element.

Force all connections to disconnect (dbDisconnect)

The dbDisconnect command forces all user connections to the specified database to be disconnected. This is useful, for example, prior to maintenance operations. Closing connections is mandatory to ensure a smooth shutdown and no data loss.

```
ttGridAdmin dbDisconnect name
                   -transactional|-immediate|-abort
                   [-nowait | -wait [timeout]]
```

No new transactions are allowed before the command executes.

A disconnection request is sent to each data instance in the grid.

See "Unloading a database from memory" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for related information.

> **Note:** The dbDisconnect command does not affect subdaemon
> connections.

Options

The dbDisconnect command has the options:

Option	Description
name	Name of the database.
-transactional|-immediate|-abort	Specify the mode for the disconnection process. You must specify one of the following modes (there is no default):
	■ **Transactional**: Allow any open transactions to be committed or rolled back before disconnecting.
	■ **Immediate**: Roll back any open transactions before immediately disconnecting.
	■ **Abort**: Abort all direct mode application processes and client/server agents in order to disconnect.
	A recommended best practice is to run dbDisconnect twice, as necessary. First run it in transactional mode. Then, after allowing some time, if not all connections have been closed yet, run it in immediate mode. Use dbStatus -connections to confirm whether connections have been closed.
	Use abort mode only as a last resort if transactional and immediate levels do not result in all connections being closed. **Abort may result in loss of data**. Abort abruptly causes every user and ttcserver process connected to the database to exit. This may result in lost transactions.
	Note: Execution in immediate mode also disconnects idle connections. Execution in transactional mode does not.

Option	Description
`-nowait` | `-wait [timeout]`	Specifies whether the command should return immediately (the default) or wait until all disconnections finish. With `-wait`, you can optionally limit the wait time to *timeout* seconds.
	Database management commands initiate a state change that is recorded in the active management instance of the grid. More specifically, the `-nowait` option causes the command to return without waiting for the state change. Use `dbDisconnectStatus` to check status of the disconnection process.
	The `-wait` option causes the command to wait for the state change to complete, when all disconnections are complete. Without *timeout* or if *timeout* is set to 0, there is no limit.
	If there is a large number of connections, it may not be advisable to use `-wait`. If you do, it is advisable to set a timeout. If disconnections have not completed in the specified wait time and you are using transactional mode, consider using immediate mode.
	Note: Even when using `-wait`, it is advisable to use `dbStatus -connections` afterward to confirm connections are closed.

Examples

This example:

1. Uses `dbStatus` to show existing connections.

2. Closes the database and confirms.

3. Disconnects in transactional mode (without wait).

4. Checks status of the `dbDisconnect` command with `dbDisconnectStatus` and the status of the connections with `dbStatus`. (The `dbDisconnect` command is in progress and the connections still exist.)

5. Disconnects in immediate mode (without wait), to be sure connections are closed.

6. Again checks status of the `dbDisconnect` command with `dbDisconnectStatus` and the status of the connections with `dbStatus`. (The `dbDisconnect` command has completed and the connections are gone.)

```
% ttGridAdmin dbStatus database1 -connections
Host       Instance  ConnId Name      Pid   Type   CHost CAddr CPid
---------- --------- ------ --------- ----- ------ ----- ----- ----
mysys5host instance3      1 database1 20233 Direct
mysys4host instance2      1 database1 26529 Direct
mysys3host instance1      1 database1  1600 Direct
mysys6host instance4      1 database1  1678 Direct

% ttGridAdmin dbClose database1
Database database1 close started

% ttGridAdmin dbStatus database1
Database database1 summary status as of Tue Nov 27 16:12:16 PST 2018

created,loaded-complete,closed
```

```
Completely created elements: 4 (of 4)
Completely loaded elements: 4 (of 4)
Completely created replica sets: 2 (of 2)
Completely loaded replica sets: 2 (of 2)

Open elements: 0 (of 4)
```

First try disconnecting in transactional mode:

```
% ttGridAdmin dbDisconnect database1 -transactional
Database database1 dbDisconnect started
```

Let some time pass, then check status—connections still exist:

```
% ttGridAdmin dbDisconnectStatus database1
Database  Host       Instance  Elem State         Started
--------- ---------- --------- ---- ------------- -----------------------
database1                           Disconnecting 2018-11-27T16:12:55.000Z
          mysys5host instance3    1 Disconnecting
          mysys4host instance2    2 Disconnecting
          mysys3host instance1    3 Disconnecting
          mysys6host instance4    4 Disconnecting

% ttGridAdmin dbStatus -connections
Database database1:

Host       Instance  ConnId Name      Pid   Type   CHost CAddr CPid
---------- --------- ------ --------- ----- ------ ----- ----- ----
mysys5host instance3      1 database1 20233 Direct
mysys4host instance2      1 database1 26529 Direct
mysys3host instance1      1 database1  1600 Direct
mysys4host instance4      1 database1  1678 Direct
```

Try again in immediate mode:

```
% ttGridAdmin dbDisconnect database1 -immediate
Database database1 dbDisconnect started
```

Check status again—now the connections are gone.:

```
% ttGridAdmin dbDisconnectStatus database1
Database  Host       Instance  Elem State        Started
--------- ---------- --------- ---- ------------ -----------------------
database1                           Complete     2018-11-27T16:14:03.000Z
          mysys5host instance3    1 Disconnected
          myshs4host instance2    2 Disconnected
          mysys3host instance1    3 Disconnected
          mysys6host instance4    4 Disconnected

% ttGridAdmin dbStatus database1 -connections
Host Instance ConnId Name Pid Type CHost CAddr CPid
---- -------- ------ ---- --- ---- ----- ----- ----
%
```

Notes

- The database must be in a closed state before you execute this command. (Closing a database does not affect existing connections, but does prevent new connections.)

- In TimesTen Scaleout, the capability to force disconnections is always enabled and the forceDisconnectEnabled connection attribute is ignored.

Check status of forced disconnection (dbDisconnectStatus)

The `dbDisconnectStatus` command reports the status of the executing or most recently executed `dbDisconnect` command.

```
ttGridAdmin dbDisconnectStatus name
```

Any of these states may be reported for the overall status of the `dbDisconnect` command:

- Defined: Disconnect has been requested but not yet initiated.

- Disconnecting: Disconnect is still in progress on at least one element.

- Failed: Disconnect failed on at least one element.

- Complete: Disconnect completed successfully on all elements.

Any of these states may be reported for the status of the disconnect on any given instance:

- Disconnecting: Disconnect is in progress on the instance.

- Failed: Disconnect failed on the instance.

- Disconnected: Disconnect completed successfully on the instance.

Options

The `dbDisconnectStatus` command has the option:

Option	Description
name	Name of the database.

Examples

A `dbDisconnectStatus` example is included in the `dbDisconnect` example in the preceding section.

Set or modify the distribution scheme of a database (dbDistribute)

The `dbDistribute` command can add, remove, evict, and replace elements of a database in the distribution map of the database, then distribute or redistribute data among elements. You must always use `-apply` to apply changes and redistribute data. You can do this either in the same command or in a separate command.

```
ttGridAdmin dbDistribute name
        [-list]
        [-add all | hostname[.instancename]]
        [-remove hostname[.instancename] [-replaceWith hostname[.instancename]]]
        [-evict hostname[.instancename] [-replaceWith hostname[.instancename]]]
        [-apply|-reset|-resync]
```

Wait until the elements are loaded on all instances on which you will perform operations before using `dbDistribute`. You can use the `dbStatus` command to confirm this.

See "Define the distribution map of the database" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for additional information.

Options

The dbDistribute command has the options:

Option	Description
name	Name of the database for data distribution changes.
-add all | *hostname*[.*instancename*]	Adds elements to the distribution map, either all currently existing elements in the grid or elements on the specified instances. (If an element was not created because an instance is down, there would be no attempt to add it with -add all. It is not currently existing.)
	Specify one instance per usage, but you can use -add more than once on a command line.
	When the additions are applied, data will be distributed evenly across the grid.
	Notes:
	■ If you use -add all, you must use -apply in the same command.
	■ Until you issue -apply, the element is marked to be added but is not actually added yet.
	Also see Notes below for this and other options taking *hostname*[.*instancename*].
-remove *hostname*[.*instancename*]	Use this option in any circumstance where you want to remove, and optionally replace, an element, such as to replace an older host system with a newer one. Also see Notes below.
	Specify one instance per usage, but you can use -remove more than once on a command line.
	It is typical to use -replaceWith to replace the element. The -remove option without -replaceWith results in redistribution of data.
	If you have a grid with k=2 and you remove one element of a replica set, you must either replace it or also remove the other element of the replica set.
	Note: Until you issue -apply, the element is marked for removal but is not actually removed yet.
	Also see "Redistributing data in a database" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Option	Description
-evict *hostname*[.*instancename*]	Use this option if all elements of a replica set (one element if k=1, two elements if k=2) have unrecoverable failures and you cannot repair them.
	Important: Using the -evict option inevitably results in data loss. Use this only as a last resort.
	Specify one instance per usage, but you can use -evict more than once on a command line.
	If you use -evict, you must evict all elements in the replica set.
	You can use -replaceWith to replace the element.
	Notes:
	■ Do not issue or apply -evict together with -add or -remove.
	■ Until you issue -apply, the element is marked for eviction but is not actually evicted yet.
	■ Eviction results in the element being forcibly unloaded.
	Also see Notes below for additional considerations.
	For additional information, see "Redistributing data in a database" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.
-replaceWith *hostname*[.*instancename*]	Optionally use this with -evict or -remove to have the specified replacement contain the same data. The element on the replacing instance must not have previously been added to the distribution.
	The -replaceWith option must immediately follow the corresponding -remove or -evict option on the command line.
-list	Displays the current and pending distribution map of the database ("Holds Data" and "Will Hold Data", respectively).
-apply	Applies the new distribution to the database. You can use this option by itself to apply settings from previous commands, or in the same command line with the settings.
-reset	Discards all distribution settings that have not yet been applied. This option cannot be combined with any other option.
	Note: You cannot use -reset while distribution (-apply) is in progress. You can try -resync instead, as appropriate.

Option	Description
`-resync`	Attempts to resynchronize metadata in the user database with metadata in the active management instance in case the state of a `dbDistribute -apply` command is unknown. For example, the user database and management instance may not have matching states due to some failure or loss of communication. In some cases, the management instance may not know about the success or failure of a `dbDistribute` operation on the data instances and is left in an intermediate state.
	This option cannot be used with any other `dbDistribute` options.
	See "Recovering from a data distribution error" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for related information.
	Note: The `-resync` option results in metadata in the management instance being read to see if there is a `dbDistribute` operation that is in progress but was neither committed nor rolled back. Resynchronizing may involve committing or rolling back the metadata changes of the `dbDistribute` operation (which are intended to be recorded in the management instance).

Examples

This example adds all elements in the grid to the distribution map then distributes data among the elements:

```
% ttGridAdmin dbDistribute database1 -add all -apply
Distribution map updated
```

You can then use the `-list` option to show the distribution map of elements in the grid (elements able to hold data):

```
% ttGridAdmin dbDistribute database11 -list
Distribution Map version: 1
RS Host        Instance  Holds Data Will Hold Data Removed Evicted
-- ----------- --------- ---------- -------------- ------- -------
 1  mysys3host griddata1      Y             Y         N       N
 1  mysys4host griddata2      Y             Y         N       N
 2  mysys5host griddata3      Y             Y         N       N
 2  mysys6host griddata4      Y             Y         N       N
```

Now remove both elements in replica set 1, then look at the `-list` output again, which indicates the two elements removed from the grid and therefore unable to hold data:

```
% ttGridAdmin dbDistribute database1 -remove mysys3host.griddata1
Element mysys3host.griddata1 is removed
Distribution map change enqueued
% ttGridAdmin dbDistribute database1 -remove mysys4host.griddata2
Element mysys4host.griddata2 is removed
Distribution map change enqueued
% ttGridAdmin dbDistribute database1 -apply
Distribution map updated

% ttGridAdmin dbDistribute database1 -list
Distribution Map version: 3
RS   Host        Instance  Holds Data Will Hold Data Removed Evicted
```

```
---- ----------- --------- ---------- -------------- ------- -------
NULL  mysys3host griddata1          N              N       Y       N
NULL  mysys4host griddata2          N              N       Y       N
   1  mysys5host griddata3          Y              Y       N       N
   1  mysys6host griddata4          Y              Y       N       N
```

The following is a new example that evicts two elements (from the same replica set) then looks at the -list output, which shows the two elements evicted from the grid and therefore unable to hold data.

```
% ttGridAdmin dbDistribute database1 -evict mysys3host.griddata1 -evict
mysys4host.griddata2 -apply
Distribution map updated

% ttGridAdmin dbDistribute database1 -list
Distribution Map version: 2
RS    Host        Instance  Holds Data Will Hold Data Removed Evicted
---- ---------- --------- ---------- -------------- ------- -------
NULL mysys3host griddata1          N              N       N       Y
NULL mysys4host griddata2          N              N       N       Y
   1 mysys5host griddata3          Y              Y       N       N
   1 mysys6host griddata4          Y              Y       N       N
```

This example shows where the -resync option successfully completed a data distribution operation:

```
% ttGridAdmin dbDistribute database1 -apply
...
```

(Process fails or is interrupted.)

```
% ttGridAdmin dbDistribute database1 -resync
Distributiom map updated
```

And this example shows where the -resync option rolled back a data distribution operation:

```
% ttGridAdmin dbDistribute database1 -apply
...
```

(Process fails or is interrupted.)

```
% ttGridAdmin dbDistribute database1 -resync
Distributiom map Rolled Back
```

Notes

- You can use -list and -resync while distribution is in progress. Other operations will fail if distribution is in progress.

- To specify an element, express its instance as *hostname*[.*instancename*]. The host name is required. The instance name is required only if there are multiple instances on the host. (See "Grid objects and object naming" on page 4-5.)

- If you need to confirm which elements are in each replica set, use the dbStatus command with the -replicaSet option.

- Once an element has been removed or evicted from the distribution, the only possibility is to eliminate it with dbDestroy -instance. It is advisable to do that as soon as possible to reclaim the disk space that it used. If you want to be able to use the instance again later, you must recreate the element with dbCreate -instance, then add it to the distribution.

- If `k=2` and one element of a replica set has an irrecoverable failure, use `-remove` and `-replaceWith` to make the replica set fully operational again. Do not use `-evict` when an active replica is available.

- If all elements of any replica set are down, you cannot perform global operations. If you cannot recover any element of the replica set, evicting the elements of the replica set will allow you to perform global operations again, but there will be permanent loss of data.

- It is valid to use `-add` instead of `-replaceWith` to replace the elements of an evicted replica set, but in either case data on the evicted replica set is lost. Also note that `-add` results in redistribution of data while `-replaceWith` (used with either `-evict` or `-remove`) does not. See "Recovering when the replica set has a permanently failed element" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for additional information.

- See "Recovering from Failure" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for additional information and considerations regarding failure modes.

List databases (dbList)

The `dbList` command lists the databases that have been created in the grid and indicates whether they have been loaded or opened.

```
ttGridAdmin dbList
```

Examples

```
% ttGridAdmin dbList
Database                    Loaded Opened
database1                      Y      Y
testdb                         Y      N
```

Load a database into memory (dbLoad)

The `ttGridAdmin dbLoad` command loads the specified database into memory. A database must be loaded and opened before it is used by applications.

```
ttGridAdmin dbLoad name
                  [-nowait | -wait [timeout]]
```

Options

The `dbLoad` command has the options:

Option	Description
name	Name of the database to load.

Option	Description
`-nowait | -wait [timeout]`	The command initiates a state change that is recorded in the active management instance of the grid.
	The `-nowait` option causes the command to return immediately without waiting for the state change. This is the default behavior.
	The `-wait` option causes the command to wait for the state change to complete, when the database element has been loaded on each instance in the grid. You can optionally subject the wait to a limit of *timeout* seconds. Otherwise, or if *timeout* is set to 0, there is no limit.
	In a large grid, it is not typical or generally advisable to use `-wait`. If you do, it is advisable to set a timeout. (See "Database management operations" on page 4-7.)

Examples

This example loads a database without waiting for the elements to be loaded on all instances, then checks status (after the database was successfully loaded):

```
% ttGridAdmin dbLoad database1
Database database1 load started
...
% ttGridAdmin dbStatus database1
Database database1 summary status as of Mon Nov 13 18:58:53 PST 2017

created,loaded,closed
Completely created elements: 4 (of 4)
Completely loaded elements: 4 (of 4)
Completely created replica sets: 0 (of 0)
Completely loaded replica sets: 0 (of 0)

Open elements: 0 (of 4)
```

Notes

- Before loading a database, it is advisable to run `dbStatus` with the `-loadReadiness` option to confirm all replica sets can be loaded.

- After the command has completed, the database is loaded but closed. (Use `dbOpen` to open it.)

- It is not necessary to run `dbLoad` after `dbCreate`, because `dbCreate` loads the database automatically.

- If you run `dbLoad` asynchronously (without waiting), you can use the `dbStatus` command to see when the database is loaded.

Open a database (dbOpen)

The `dbOpen` command opens the database so that applications can connect to it.

```
ttGridAdmin dbOpen name
                    [-nowait | -wait [timeout]]
```

Options

The `dbOpen` command has the options:

Option	Description
name	Name of the database to open.
-nowait | -wait [*timeout*]	The command initiates a state change that is recorded in the active management instance of the grid.
	The -nowait option causes the command to return immediately without waiting for the state change. This is the default behavior.
	The -wait option causes the command to wait for the state change to complete, when the database element has been opened on each instance in the grid. You can optionally subject the wait to a limit of *timeout* seconds. Otherwise, or if *timeout* is set to 0, there is no limit.
	In a large grid, it is not typical or generally advisable to use -wait. If you do, it is advisable to set a timeout. (See "Database management operations" on page 4-7.)

Examples

This example opens a database without waiting for the elements to be opened on all instances, then checks status (after the database was opened successfully):

```
% ttGridAdmin dbOpen database1
Database database1 open started
...
% ttGridAdmin dbStatus database1
Database database1 summary status as of Mon Nov 13 19:24:39 PST 2017

created,loaded-complete,open
Completely created elements: 4 (of 4)
Completely loaded elements: 4 (of 4)
Completely created replica sets: 2 (of 2)
Completely loaded replica sets: 2 (of 2)

Open elements: 4 (of 4)
```

Notes

- The database must be loaded or loading (performed automatically by dbCreate).

- The database must have a distribution map (dbDistribute -apply).

- If you run dbOpen asynchronously (without waiting), you can use the dbStatus command to see when the database is open.

- If a database has been opened with dbOpen, attempting to open it again typically results in an error. However, if any element is in "open failed" state, you can retry dbOpen. Doing so will change any element in "open failed" state to "loaded" state, which will result in TimesTen Scaleout trying to open it again.

Monitor the status of a database (dbStatus)

The dbStatus command reports the status of a database or databases or the status of specified components of the database or databases, using information from the active management instance. This includes the status of any pending command to create, destroy, load, unload, open, or close the database. You can also request additional details, or request information about the state of each instance regarding whether its element can be loaded.

```
ttGridAdmin dbStatus [name]
```

```
[-summary]
[-element]
[-replicaSet]
[-dataSpaceGroup]
[-all]
[-details]
[-loadReadiness]
[-epochs]
[-connections [-proxy] [-system]]
```

You can also refer to dbStatus discussion and examples in "Recovering from Failure" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Options

The dbStatus command has the options:

Option	Description
name	Name of the database for which to display status. The default is to display status of all databases in the grid.
-summary	Shows an overall summary of database status. (This is the default mode.)
-element	Shows the status of each element of the database.
-replicaSet	Shows the status of each replica set of the database.
-dataSpaceGroup	Shows the status of each data space group of the database.
-all	Shows summary, element, replica set, and data space group status (equivalent to -summary -element -replicaSet -dataSpaceGroup).
-details	Shows daemon state information in addition to the status information from the active management instance. You can use this option in addition to any of the preceding options.
-loadReadiness	Shows information, including up/down status, indicating whether instances in each replica set are in a state where their elements can be loaded. It is advisable to use this option before trying to load a database. You can also use it while a load is in progress.
-epochs	Shows the most recent epochs available for each element of the grid, and the most recent epoch that could be used for recovery. An epoch is a transaction that marks a globally consistent point in time across all elements of the database. See "Epoch transactions" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.
-connections	Displays information for existing connections to the specified database. Without the -proxy or -system suboptions, only application connections are shown.
-proxy	Used with the -connections option, this also displays information for all proxy connections associated with existing application connections. **Note**: This option cannot be used without the -connections option.
-system	Used with the -connections option, this also displays TimesTen internal connections, such as those used by subdaemons and TimesTen utilities. **Note**: This option cannot be used without the -connections option.

Overall database status

The `dbStatus` command indicates the status of the database as a whole with a line showing overall created/destroyed, loaded/unloaded, and opened/closed states. (For example: `"created,loaded-complete,closed"`.)

The states of `created`, `creating`, `destroyed`, `loading`, `loaded`, `unloaded`, `unloading`, `opening`, `opened`, `closing`, and `closed` indicate that the corresponding database management command is in progress or has finished, as stated.

In addition:

- `created-partial` or `creating-partial`: Some elements of the database are in the process of being created or have successfully been created, but others could not be created.

- `createFailed`: Creation of the database failed. This occurs when no elements could be created, such as when every TimesTen instance is down.

- `loaded-partial` or `loading-partial`: The `dbDistribute` command has not yet been executed on the database (so no replica sets have been defined) and at least one element could not be created or loaded.

- `loaded-incomplete` or `loading-incomplete`: At least one replica has no elements that finished loading successfully.

- `loaded-functional` or `loading-functional`: At least one element from each replica set is loaded.

- `loaded-complete` or `loading-complete`: Every element loaded successfully.

- `notLoaded`: Loading of the database failed—none of the elements is loaded or loading.

These states can help you determine if the grid is usable even if it is not fully operational. For example, you can execute `dbOpen` before all the elements have been loaded.

Element status values

The `dbStatus` command returns these database element status values:

> **Note:** See "Troubleshooting distributed transactions" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for recommendations regarding these status values.

Status	Description
close failed	The attempt to close the element failed.
closing	The element is in the process of closing.
create failed	The attempt to create the element failed.
creating	The element is being created.
destroy failed	The attempt to destroy the element failed.
destroyed	The element has been destroyed.
destroying	The element is being destroyed.
down	The data instance where this element is located is not running.

Status	Description
evicted	The element was evicted or removed through dbDistribute and has been unloaded from RAM.
evicted (loaded)	The element was evicted or removed through dbDistribute but unloading it from RAM has not yet begun.
evicted (unloading)	The element was evicted or removed through dbDistribute and is being unloaded from RAM.
load failed	The attempt to load the element into RAM failed.
loaded	The element is loaded into RAM.
loading	The element is being loaded into RAM.
opened	The element is open.
open failed	The attempt to open the element failed.
opening	The element is in the process of opening.
uncreated	The element should be created, but creation has not yet started.
unloaded	The element has been unloaded from RAM.
unloading	The element is being unloaded from RAM.
waiting for seed	The element will be loaded into RAM, but not until after the other element in its replica set is loaded.

Connections status

This section describes information displayed by the -connections, -proxy, and -system options that show existing connections.

Connection status item	Description
Host	For the target of the connection, name of the host object in the model for the host where the data instance resides.
Instance	For the target of the connection, name of the instance object in the model for the data instance.
ConnId	Connection ID of the connection to the data instance.
Name	Name of the connection as indicated by the TimesTen ConnectionName connection attribute. (See "ConnectionName" on page 2-54 for information about that attribute.)
Pid	Operating system process ID of the process that established the connection. For direct mode applications, this is the process ID of the application. For client/server applications, this is the process ID of the client/server ttcserver process acting on behalf of the application.

Connection status item	Description
Type	The type of connection. One of the following:
	■ `Direct` for connections from direct mode applications.
	■ `C/S` for connections from client/server applications.
	■ `Proxy` for connections created by TimesTen that work on behalf of application connections, such as a connection to a different grid element that is necessary to access some of the data.
	■ `GCW` for TimesTen internal connections from grid connection workers.
	■ `Subdaemon` for TimesTen internal connections from TimesTen subdaemons.
	■ `TTStats` for TimesTen internal connections for collection of statistics.
CHost	For client/server connections, the host name of the client where the application is running.
	Note: This item is not shown when the -proxy option is used.
CAddr	For client server connections, the IP address of the client where the application is running.
	Note: This item is not shown when the -proxy option is used.
CPid	For client/server connections, the operating system process ID of the application.
	Note: This item is not shown when the -proxy option is used.
PHost	For proxy connections, the name of the host where the proxy connection is established.
PInstance	For proxy connections, the name of the TimesTen instance where the proxy connection is established.
PPid	For proxy connections, the operating system process ID of the process that established the connection.
PConnId	For proxy connections, the connection ID.

Examples

Database status examples

Key for these examples:

- **RS**: Identifying number of the replica set that each element belongs to.

- **DS**: Identifying number of the data space group that each element belongs to.

- **Elem**: Element number for each element.

- **Status**: Status of the operation on each element. See "Status values" above for the list of element states that can be returned.

This example shows complete dbStatus output after a database has had its distribution specified, but the database is closed.

```
% ttGridAdmin dbStatus database1 -all
Database database1 summary status as of Thu Nov 17 13:28:16 PST 2016

created,loaded-complete,closed
Completely created elements: 4 (of 4)
Completely loaded elements: 4 (of 4)
Completely created replica sets: 2 (of 2)
```

```
Completely loaded replica sets: 2 (of 2)

Open elements: 0 (of 4)

Database database1 element level status as of Thu Nov 17 13:28:16 PST 2016

Host        Instance     Elem Status Date/Time of Event  Message
----------- ------------ ---- ------ ------------------- -------
 mysys3host griddata1       3 loaded 2016-11-16 17:36:39
 mysys4host griddata2       1 loaded 2016-11-16 17:36:40
 mysys5host griddata3       4 loaded 2016-11-16 17:36:39
 mysys6host griddata4       2 loaded 2016-11-16 17:36:41

Database database1 Replica Set status as of Thu Nov 17 13:28:16 PST 2016

RS DS Elem Host        Instance  Status Date/Time of Event  Message
-- -- ---- ----------- --------  ------ ------------------ -------
 1  1    3 mysys3host  griddata1 loaded 2016-11-16 17:36:39
    2    1 mysys4host  griddata2 loaded 2016-11-16 17:36:40
 2  1    4 mysys5host  griddata3 loaded 2016-11-16 17:36:39
    2    2 mysys6host  griddata4 loaded 2016-11-16 17:36:41

Database database1 Data Space Group status as of Thu Nov 17 13:28:16 PST 2016

DS RS Elem Host        Instance  Status Date/Time of Event  Message
-- -- ---- ----------- --------- ------ ------------------ -------
 1  1    3 mysys3host  griddata1 loaded 2016-11-16 17:36:39
    2    4 mysys5host  griddata3 loaded 2016-11-16 17:36:39
 2  1    1 mysys4host  griddata2 loaded 2016-11-16 17:36:40
    2    2 mysys6host  griddata4 loaded 2016-11-16 17:36:41
```

This example shows load readiness with all instances up, then with one instance in a replica set down, then with both instances in a replica set down. If all instances in a replica set are down, the database cannot be loaded.

```
% ttGridAdmin dbStatus database1 -loadReadiness
Data Elements:
RS DS Instance            State
-- -- ------------------- --------
 1  1 mysys3host.griddata1 Unloaded
 1  2 mysys4host.griddata2 Unloaded
 1                         Loadable
 2  1 mysys5host.griddata3 Unloaded
 2  2 mysys6host.griddata4 Unloaded
 2                         Loadable

database1 load state: Loadable
Total Elements Loaded:0/4
...

% ttGridAdmin dbStatus database1 -loadReadiness
Data Elements:
RS DS Instance            State
-- -- ------------------- --------
 1  1 mysys3host.griddata1 Down
 1  2 mysys4host.griddata2 Unloaded
 1                         Loadable
 2  1 mysys5host.griddata3 Unloaded
 2  2 mysys6host.griddata4 Unloaded
 2                         Loadable
```

```
            database1 load state: Loadable
            Total Elements Loaded:0/4
            ...

            % ttGridAdmin dbStatus database1 -loadReadiness
            Data Elements:
            RS DS Instance              State
            -- -- ------------------- ------------
             1  1 mysys3host.griddata1 Down
             1  2 mysys4host.griddata2 Down
             1                         Not Loadable
             2  1 mysys5host.griddata3 Unloaded
             2  2 mysys6host.griddata4 Unloaded
             2                         Loadable

            database1 load state: Not Loadable
            Total Elements Loaded:0/4
```

This example shows the epochs of the database. The important point is that if durability=0 and no recovery epoch is shown, the database is not recoverable.

```
% ttGridAdmin dbStatus database1 -epochs
Database database1 element level status as of Tue Jan  9 16:49:39 PST 2018

Host        Instance  Elem Status Recent Epochs
---------- --------- ---- ------ --------------------------------------------------------------
mysys4host griddata2    1 loaded 286.3 288.1 290.2 292.4 294.3 296.1 298.2 300.1 302.2 304.4
mysys6host griddata4    2 loaded 286.3 288.1 290.2 292.4 294.3 296.1 298.2 300.1 302.2 304.4
mysys3host griddata1    3 loaded 286.3 288.1 290.2 292.4 294.3 296.1 298.2 300.1 302.2 304.4
mysys5host griddata3    4 loaded 286.3 288.1 290.2 292.4 294.3 296.1 298.2 300.1 302.2 304.4

Most recent recovery epoch: 304.4
```

Connection status examples

Examples are shown for the -connections option by itself, -connections with -proxy, -connections with -system, and -connections with both -proxy and -system.

```
% ttgridadmin dbstatus database1 -connections
Host    Instance ConnId Name        Pid  Type   CHost  CAddr         CPid
------ --------- ------ ----------- ---- ------ ------ ------------- ----
mysys1 instance1      1 database1   8631 Direct
mysys1 instance1      2 con1        8631 Direct
mysys1 instance1      3 con2        8631 Direct
mysys2 instance2      1 database1cs 8653 C/S    mysys2 10.90.137.240 8637
mysys2 instance2      2 con1        8666 C/S    mysys2 10.90.137.240 8637

% ttgridadmin dbstatus database1 -connections -proxy
Host    Instance ConnId Name        Pid  Type   PHost  PInstance PPid  PConnId
------ --------- ------ ----------- ---- ------ ------ --------- ----- -------
mysys1 instance1      1 database1   8631 Direct
mysys1 instance1      2 con1        8631 Direct
mysys1 instance1      2 con1        8631 Proxy  mysys2 instance2 31210       4
mysys1 instance1      3 con2        8631 Direct
mysys1 instance1      3 con2        8631 Proxy  mysys2 instance2 31210       3
mysys2 instance2      1 database1cs 8653 C/S
mysys2 instance2      2 con1        8666 C/S

% ttgridadmin dbstatus database1 -connections -system
Host    Instance ConnId Name                             Pid  Type    CHost   CAddr         CPid
```

```
------ --------- ------ ------------------------ ----- --------- -------- ------------- ----
mysys1 instance1      1 database1                 8631 Direct
mysys1 instance1      2 con1                      8631 Direct
mysys1 instance1      3 con2                      8631 Direct
mysys1 instance1    128 Grid Epoch Generator(TM=2) 31183 GCW
mysys1 instance1    129 ttStats Collector        31183 GCW
mysys1 instance1    130 ttStats Collector        31871 TTStats
mysys1 instance1    131 Garbage Collector        30876 Subdaemon
mysys1 instance1    132 Grid Watch Remote TM     30876 Subdaemon
mysys1 instance1    133 Grid Rem Elem Mon        30876 Subdaemon
mysys1 instance1    134 XactId Rollback          30876 Subdaemon
mysys1 instance1    135 Grid Epoch Generator     30876 Subdaemon
mysys1 instance1    136 Grid Seq Batch           30876 Subdaemon
mysys1 instance1    137 GCW Watcher              30876 Subdaemon
mysys1 instance1    138 HistGC                   30876 Subdaemon
mysys1 instance1    139 Log Marker               30876 Subdaemon
mysys1 instance1    140 IndexGC                  30876 Subdaemon
mysys1 instance1    141 Grid Task                30876 Subdaemon
mysys1 instance1    142 Deadlock Detector        30876 Subdaemon
mysys1 instance1    143 Flusher                  30876 Subdaemon
mysys1 instance1    144 Monitor                  30876 Subdaemon
mysys1 instance1    145 Checkpoint               30876 Subdaemon
mysys1 instance1    146 Rollback                 30876 Subdaemon
mysys1 instance1    147 Manager                  30876 Subdaemon
mysys2 instance2      1 database1cs               8653 C/S       mysys2 10.90.137.240   8637
mysys2 instance2      2 con1                      8666 C/S       mysys2 10.90.137.240   8637
mysys2 instance2      3 con2                     31210 GCW
mysys2 instance2      4 con1                     31210 GCW
mysys2 instance2    128 Grid Epoch Generator(TM=1) 31210 GCW
mysys2 instance2    129 ttStats Collector        31210 GCW
mysys2 instance2    130 ttStats Collector        31878 TTStats
mysys2 instance2    131 Grid Watch Remote TM     30950 Subdaemon
mysys2 instance2    132 HistGC                   30950 Subdaemon
mysys2 instance2    133 GCW Watcher              30950 Subdaemon
mysys2 instance2    134 Grid Epoch Generator     30950 Subdaemon
mysys2 instance2    135 Grid Seq Batch           30950 Subdaemon
mysys2 instance2    136 XactId Rollback          30950 Subdaemon
mysys2 instance2    137 Garbage Collector        30950 Subdaemon
mysys2 instance2    138 Grid Task                30950 Subdaemon
mysys2 instance2    139 Log Marker               30950 Subdaemon
mysys2 instance2    140 Grid Rem Elem Mon        30950 Subdaemon
mysys2 instance2    141 Flusher                  30950 Subdaemon
mysys2 instance2    142 IndexGC                  30950 Subdaemon
mysys2 instance2    143 Checkpoint               30950 Subdaemon
mysys2 instance2    144 Deadlock Detector        30950 Subdaemon
mysys2 instance2    145 Monitor                  30950 Subdaemon
mysys2 instance2    146 Rollback                 30950 Subdaemon
mysys2 instance2    147 Manager                  30950 Subdaemon

% ttgridadmin dbstatus database1 -connections -proxy -system
Host   Instance  ConnId Name                     Pid   Type      PHost  PInstance PPid  PConnId
------ --------- ------ ------------------------ ----- --------- ------ --------- ----- -------
mysys1 instance1      1 database1                 8631 Direct
mysys1 instance1      2 con1                      8631 Direct
mysys1 instance1      2 con1                      8631 Proxy     mysys2 instance2 31210       4
mysys1 instance1      3 con2                      8631 Direct
mysys1 instance1      3 con2                      8631 Proxy     mysys2 instance2 31210       3
mysys1 instance1    128 Grid Epoch Generator(TM=2) 31183 GCW
mysys1 instance1    129 ttStats Collector        31183 GCW
mysys1 instance1    130 ttStats Collector        31871 TTStats
```

```
mysys1 instance1    130 ttStats Collector        31871 Proxy     mysys2 instance2 31210    129
mysys1 instance1    131 Garbage Collector        30876 Subdaemon
mysys1 instance1    132 Grid Watch Remote TM     30876 Subdaemon
mysys1 instance1    133 Grid Rem Elem Mon        30876 Subdaemon
mysys1 instance1    134 XactId Rollback          30876 Subdaemon
mysys1 instance1    135 Grid Epoch Generator     30876 Subdaemon
mysys1 instance1    135 Grid Epoch Generator     30876 Proxy     mysys2 instance2 31210    128
mysys1 instance1    136 Grid Seq Batch           30876 Subdaemon
mysys1 instance1    137 GCW Watcher              30876 Subdaemon
mysys1 instance1    138 HistGC                   30876 Subdaemon
mysys1 instance1    139 Log Marker               30876 Subdaemon
mysys1 instance1    140 IndexGC                  30876 Subdaemon
mysys1 instance1    141 Grid Task                30876 Subdaemon
mysys1 instance1    142 Deadlock Detector        30876 Subdaemon
mysys1 instance1    143 Flusher                  30876 Subdaemon
mysys1 instance1    144 Monitor                  30876 Subdaemon
mysys1 instance1    145 Checkpoint               30876 Subdaemon
mysys1 instance1    146 Rollback                 30876 Subdaemon
mysys1 instance1    147 Manager                  30876 Subdaemon
mysys2 instance2      1 database1cs               8653 C/S
mysys2 instance2      2 con1                      8666 C/S
mysys2 instance2      3 con2                     31210 GCW
mysys2 instance2      4 con1                     31210 GCW
mysys2 instance2    128 Grid Epoch Generator(TM=1) 31210 GCW
mysys2 instance2    129 ttStats Collector        31210 GCW
mysys2 instance2    130 ttStats Collector        31878 TTStats
mysys2 instance2    130 ttStats Collector        31878 Proxy     mysys1 instance1 31183    129
mysys2 instance2    131 Grid Watch Remote TM     30950 Subdaemon
mysys2 instance2    132 HistGC                   30950 Subdaemon
mysys2 instance2    133 GCW Watcher              30950 Subdaemon
mysys2 instance2    134 Grid Epoch Generator     30950 Subdaemon
mysys2 instance2    134 Grid Epoch Generator     30950 Proxy     mysys1 instance1 31183    128
mysys2 instance2    135 Grid Seq Batch           30950 Subdaemon
mysys2 instance2    136 XactId Rollback          30950 Subdaemon
mysys2 instance2    137 Garbage Collector        30950 Subdaemon
mysys2 instance2    138 Grid Task                30950 Subdaemon
mysys2 instance2    139 Log Marker               30950 Subdaemon
mysys2 instance2    140 Grid Rem Elem Mon        30950 Subdaemon
mysys2 instance2    141 Flusher                  30950 Subdaemon
mysys2 instance2    142 IndexGC                  30950 Subdaemon
mysys2 instance2    143 Checkpoint               30950 Subdaemon
mysys2 instance2    144 Deadlock Detector        30950 Subdaemon
mysys2 instance2    145 Monitor                  30950 Subdaemon
mysys2 instance2    146 Rollback                 30950 Subdaemon
mysys2 instance2    147 Manager                  30950 Subdaemon
```

Unload a database (dbUnload)

The dbUnload command unloads the specified database from memory.

```
ttGridAdmin dbUnload name
                  [-nowait | -wait [timeout]]
                  [-force]
```

Important: If a dbUnload command is issued while a transaction is in progress, the command will not wait for the transaction to complete. Data may be lost as a result.

Options

The `dbUnload` command has the options:

Option	Description
name	Name of the database to unload.
-nowait | -wait [*timeout*]	The command initiates a state change that is recorded in the active management instance of the grid.
	The -nowait option causes the command to return immediately without waiting for the state change. This is the default behavior.
	The -wait option causes the command to wait for the state change to complete, when the database element has been unloaded on each instance in the grid. You can optionally subject the wait to a limit of *timeout* seconds. Otherwise, or if *timeout* is set to 0, there is no limit.
	In a large grid, it is not typical or generally advisable to use -wait. If you do, it is advisable to set a timeout. (See "Database management operations" on page 4-7.)
-force	If `Durability=0` and at least one replica set is completely down, this option allows the unload to proceed anyway.
	Important: Using this option will likely result in data loss.
	(Normally, to prevent data loss, a database with `Durability=0` cannot be unloaded unless at least one element from every replica set is loaded.)

Examples

This example unloads a database without waiting for the elements to be unloaded on all instances, then checks status (after the database was successfully unloaded).

```
% ttGridAdmin dbUnload database1
Database database1 unload started
...
% ttGridAdmin dbStatus database1
Database database1 summary status as of Mon Nov 13 18:52:47 PST 2017

created,unloaded,closed
Completely created elements: 4 (of 4)
Completely loaded elements: 0 (of 4)
Completely created replica sets: 0 (of 0)
Completely loaded replica sets: 0 (of 0)

Open elements: 0 (of 4)
```

Notes

- Do not begin any transactions after issuing a `dbUnload` command.

- All connections to the database must be closed.

- The database must be closed.

- If you run `dbUnload` asynchronously (without waiting), you can use the `dbStatus` command to see when the database is loaded.

Grid operations

Use the `ttGridAdmin` commands in this section to create a grid in the model, configure passwordless SSH for the grid, gather information about the grid, and make changes to the grid. There is also a command to produce a `sys.odbc.ini` file for use by clients outside of the grid.

Export sys.odbc.ini for client/server connections outside grid (gridClientExport)

The `gridClientExport` command produces a `sys.odbc.ini` file that can be used by TimesTen instances that are not part of the grid to access databases in the grid.

```
ttGridAdmin gridClientExport [filepath]
```

The resulting file contains definitions of all client/server connectables defined in the grid. You must manually copy this file to any TimesTen client instances outside of the grid from which you want to connect to databases in the grid.

Options

The `gridClientExport` command has the option:

Option	Description
filepath	Path and name of the file where the `sys.odbc.ini` entries are written.
	If no file is specified, the entries are output to `stdout`.

Examples

This example exports the `sys.odbc.ini` entries to the file `sys_export.odbc.ini`, then shows the contents of that file.

```
% ttGridAdmin gridClientExport /sw/tten/grid/clients/sys_export.odbc.ini
% cd /sw/tten/grid/clients
% more sys_export.odbc.ini
[ODBC Data Sources]
database1client=TimesTen 18.1 Client Driver

[database1client]
TTC_SERVER_DSN=database1
# External address/port info for mysys3host.instance1
TTC_SERVER1=mysys3.example.com/21000
# External address/port info for mysys4host.instance1
TTC_SERVER2=mysys4.example.com/21000
ConnectionCharacterSet=AL32UTF8
UID=ttclient
```

Notes

This command uses the external address of the host.

Create a grid (gridCreate)

The `gridCreate` command creates a grid and the initial version of the model.

```
ttGridAdmin gridCreate name
                       -k n
                       -membershipConfig filepath
```

```
[-address addr]
[-internalAddress addr]
[-externalAddress addr]
[-mgmtPort n]
[-host name]
[-retainDays numdays]
[-retainVersions n]
[-warnThresh percent]
[-noDataSpaceGroup]
[-walletDir path]
```

The instance from which the command is run becomes the initial management instance of the new grid. Additional instances (data instances and a second management instance) can then be created and joined to the grid later.

Options

The `gridCreate` command has the options:

Option	Description
name	Specifies the name for the grid in the model.
-k *n*	Specifies the degree of K-safety that this grid provides. Specify a value of 1 or 2.
-membershipConfig *filepath*	Path and name of the membership client configuration file, which contains the host name and port of each membership server.
	The contents of this file will be automatically provisioned in every instance in the grid.
	Sample contents:
	`Servers zk1.example.com!2181,zk2.example.com!2181,` `zk3.example.com!2181`
	Note: Either colons or exclamation marks can be used between host and port. (Always use exclamation marks with IPv6 addresses, which themselves include colons.)
	Also see "Membership operations" on page 4-97 for information about exporting or importing the membership client configuration file and "Configure a grid as a membership service client" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for additional information.
-internalAddress *addr*	DNS name or IP address of the local system for internal communications, inside the grid. Use this together with -externalAddress.
	This option takes one name or address only, and a specified name must resolve to one IP address or to multiple IP addresses on the same network segment.
	If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid.
	Also see Notes below and "Address formats" on page 4-6.

Option	Description
-externalAddress *addr*	DNS name or IP address of the local system for external communications, outside the grid, for client/server connections. Use this together with -internalAddress.
	This option takes one name or address only, but a name may resolve to one or more IP addresses.
	If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid.
	Also see Notes below and "Address formats" on page 4-6.
-address *addr*	DNS name or IP address of the local system for both external and internal communications, if a single address is used. Setting -address *xxx* is exactly equivalent to setting -internalAddress *xxx* and -externalAddress *xxx*.
	This option takes one name or address only, and a specified name must resolve to one IP address or to multiple IP addresses on the same network segment.
	If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid.
	Note: Using a single address is not recommended for production environments.
	Also see Notes below and "Address formats" on page 4-6.
-mgmtPort *n*	Port number used by the initial management instance for replication when management data on the active management instance is replicated. This is required if there will be two management instances. The default is 3754.
-host *name*	Specifies the name that will be given to the host object in the model for the initial host in the grid. If not specified, the first component of the operating system host name is used (the host name up to but not including the first ".", such as myhost).
-retainDays *numdays*	Specifies that old versions of the model should be retained for *numdays* days, then automatically deleted. If *numdays* is 0, then old versions of the model are not automatically deleted based on their age. The default is 30.
	Also see Notes below.
-retainVersions *n*	Specifies that *n* old versions of the model should be retained. Anything older than the newest *n* versions are deleted. If *n* is 0, then old versions of the model are not automatically deleted based on the number of versions. The default is 10.
	Also see Notes below.
-warnThresh *percent*	Management instances store metadata for the grid and model. If the metadata on the active management instance fills beyond this percentage of capacity, ttGridAdmin commands result in warnings. The default is 90% full.
-noDataSpaceGroup	Specifies that the initial host in the grid is not assigned to a data space group. If this option is not specified, the first host is assigned to data space group 1.
	Do not set this option if the first host will contain a data instance as well as the management instance.

Option	Description
-walletDir *path*	For the first management instance of the grid being created, path to the directory where the Oracle Wallet with cryptographic information will be stored. The default is *timesten_home*/info.
	Wallets for multiple instances can be stored in the same directory, which can be shared between the instances, such as through NFS.

Examples

```
% ttGridAdmin gridCreate grid1 -k 2 -membershipConfig
/sw/tten/grid/zkcfg/membership.conf -internalAddress intmysys1.example.com
-externalAddress extmysys1.example.com -host mysys1host
Grid grid1 created
```

Notes

- You cannot execute this command from an instance that is or has ever been part of another grid.

- You cannot retry gridCreate if it fails. You must remove and recreate the management instance with ttInstanceDestroy and ttInstanceCreate. See "Destroying a grid" and "Creating the initial management instance" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for examples. See "ttInstanceDestroy" on page 5-62 and "ttInstanceCreate" on page 5-59 for reference information.

- Hosts in the grid may be configured with either one or two network addresses, depending on system topology. If configured with two addresses, one is used for communications with systems inside the grid (internal) and one is used for client/server access to databases inside the grid from systems outside the grid (external). If configured with one address, which is not recommended for production environments, it is used for both internal and external communications. You must either set -address or set -internalAddress and -externalAddress.

- You can specify both -retainDays and -retainVersions, in which case old versions of the model are automatically deleted if they are older than *numdays* days old *and* there are more than *n* old versions. If one option is specified as zero, then only the other option takes effect. If both are zero, old versions of the model are not automatically deleted.

- Creating a grid creates version 1 of the grid model.

Display information about the grid (gridDisplay)

Use the gridDisplay command to display information about the grid.

```
ttGridAdmin gridDisplay
```

Examples

```
Grid name:              grid1
Grid GUID:              9D049059-1BF2-47E4-AEFA-D3ABA03F609E
Created:                2017-10-30 19:05:47.000000
Major Release:          18.1
Created Release:        18.1.4.1.0
K:                      2
Admin Userid:           ttuser1
Admin UID:              126
```

```
Admin Group:              timesten
Admin GID:                59031
Retain Days:              30
Retain Versions:          10
Warn Threshold:           90
Perm In Use Pct:          8
Temp In Use Pct:          14
```

Get diagnostic information about the grid (gridDump)

The `gridDump` command outputs diagnostic information about the grid to the specified file. This command outputs a very large amount of information and is intended for use by Oracle Support.

```
ttGridAdmin gridDump [filepath]
```

Options

The `gridDump` command has the option:

Option	Description
filepath	Path and name of the file where diagnostic information is written.
	If no file is specified, the information is written to `stdout`.

Examples

This example outputs to the file `griddumpout`. (When the dump goes to a file, the command has no visible output.)

```
% ttGridAdmin gridDump /sw/tten/grid/misc/griddumpout
```

Collect log information about the grid (gridLogCollect)

The `gridLogCollect` command collects daemon logs and other diagnostic information along with TimesTen configuration files from all instances in the grid. The aggregation of all of this is a collection.

```
ttGridAdmin gridLogCollect -repository reponame
                           [collection]
```

Options

The `gridLogCollect` command has the options:

Option	Description
-repository *reponame*	Name of the repository where the collection of logs, diagnostic information, and configuration files is stored.
	See "Create a repository (repositoryCreate)" on page 4-111.
collection	Name of the collection created to store the logs, diagnostic information, and configuration files.
	If not specified, the name will be a timestamp in the format *Lyyyymmddhhmmss*.

Examples

This example creates a repository then creates a collection of logs, diagnostic information, and configuration files in that repository. (See "Create a repository

(repositoryCreate)" on page 4-111 for information about the repositoryCreate command.)

```
% ttGridAdmin repositoryCreate repocollection -path /repositories
-method scp -address mysys1.example.com
Repository repocollection created

% ttGridAdmin gridLogCollect -repository repocollection mycollection
Logs copied to collection mycollection in repository repocollection
```

In the repocollection directory, the repository.json file has information about the repository.

The mycollection directory contains logs and configuration files for each instance. (See "Collecting grid logs" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for information about the log files.)

Notes

These are automatically included in the collection:

- Contents of the diag directory on each instance (or other diagnostics directory according to the supportlog setting in timesten.conf), such as daemon logs and core files

- TimesTen configuration files from the conf directory on each instance.

- Any .inval files from the DataStore directory of each element, as specified in the database definition

Modify grid settings (gridModify)

The gridModify command modifies properties of the grid, such as how long previous models of the grid will be retained or how many previous models of the grid will be retained.

```
ttGridAdmin gridModify [-retainDays numdays]
                       [-retainVersions n]
                       [-warnThresh percent]
```

Options

The gridModify command has the options:

Option	Description
-retainDays *numdays*	Specifies that old versions of the model should be retained for *numdays* days, then automatically deleted. If *numdays* is 0, then old versions of the model are not automatically deleted based on their age. The default is 30. Also see Notes below.
-retainVersions *n*	Specifies that *n* old versions of the model should be retained. Anything older than the newest *n* versions are deleted. If *n* is 0, then old versions of the model are not automatically deleted based on the number of versions. The default is 10. Also see Notes below.
-warnThresh *percent*	Management instances store metadata for the grid and model. If the metadata on the active management instance fills beyond this percentage of capacity, ttGridAdmin commands result in warnings. The default is 90% full.

Examples

This example shows selected output from `gridDisplay` before and after executing `gridModify` to change the number of days to retain old versions of the model.

```
% ttGridAdmin gridDisplay
Grid name:              grid1
...
Retain Days:            30
Retain Versions:        10
...

% ttGridAdmin gridModify -retainDays 20
Grid Definition modified.

% ttGridAdmin gridDisplay
Grid name:              grid1
...
Retain Days:            20
Retain Versions:        10
...
```

Notes

You can specify both `-retainDays` and `-retainVersions`, in which case old versions of the model are automatically deleted if they are older than *numdays* days old *and* there are more than *n* old versions. If one option is specified as zero, then only the other option takes effect. If both are zero, old versions of the model are not automatically deleted.

Configure SSH (gridSshConfig)

The `gridSshConfig` command configures a set of TimesTen Scaleout hosts for passwordless SSH connection, as needed or as specified.

```
ttGridAdmin gridSshConfig [ [-mgmtAddress addr1 [addr2]]
                            [-dataAddress addr1 [addr2 [addr3...]]]
                            [-repoAddress addr1 [addr2 [addr3...]]] ] |
                            [-internalAddress addr1 [addr2 [addr3...]]]
```

Either use the `-mgmtAddress` option, `-dataAddress` option, and `-repoAddress` option (as applicable) or use the `-internalAddress` option, which cannot be used with any other option. Each address can be an IPv4 address, an IPv6 address, or (typically) a DNS name. Also see "Address formats" on page 4-6.

You are prompted for the operating system password of the operating system user executing the command. That user must exist with the same password, UID, and group membership on every host to be configured.

Choose one of these modes of operation for the `gridSshConfig` command:

- Run `ttGridAdmin` from outside a TimesTen instance, where `TIMESTEN_HOME` is *not* set, using the `-mgmtAddress` option (to specify management instance hosts), the `-dataAddress` option (to specify data instance hosts), and, as needed, the `-repoAddress` option (to specify repository hosts). Run `ttGridAdmin` from the TimesTen installation `bin` directory in this case. Passwordless SSH will be configured between hosts only as needed for TimesTen Scaleout to function.

- Run `ttGridAdmin` from inside a TimesTen instance, where `TIMESTEN_HOME` is set. None of the options is necessary in this case. TimesTen determines from the grid model what each host is used for (management, data, or repository) and

configures passwordless SSH between hosts only as needed for TimesTen Scaleout to function.

- Run ttGridAdmin from outside a TimesTen instance, where TIMESTEN_HOME is *not* set, using the -internalAddress option to specify all-to-all passwordless SSH between all specified hosts, regardless of how the hosts are used (management, data, or repository). Run ttGridAdmin from the TimesTen installation bin directory in this case, but this mode of operation is NOT recommended, for security reasons.

After the gridSshConfig command is executed by a user, that user should be able to connect between hosts through SSH as needed without specifying a password (for example, between management hosts or from management hosts to data hosts). The ttGridAdmin utility will confirm this in its output after execution of the command.

> **Note:** You may choose to manually configure passwordless SSH between the hosts of your grid, as needed, without using gridSshConfig.

Options

The gridSshConfig command has the options:

Option	Description
-mgmtAddress *addr1* [*addr2*]	Addresses of hosts with management instances to configure for passwordless SSH access, as necessary.
-dataAddress *addr1* [*addr2* [*addr3...*]]	Addresses of hosts with data instances to configure for passwordless SSH access, as necessary.
-repoAddress *addr1* [*addr2* [*addr3...*]]	Addresses of hosts with repositories to configure for passwordless SSH access, as necessary.
-internalAddress *addr1* [*addr2* [*addr3...*]]	Addresses of hosts to configure for all-to-all passwordless SSH access.
	Use of this option is NOT recommended, for security reasons.
	You cannot use this option with any other option.

Examples

This example is run on mysys1.example.com, outside of any TimesTen instance, from the installation bin directory. It is run for four hosts (two management and two data).

```
% ./ttGridAdmin gridSshConfig -mgmtAddress mysys1.example.com mysys2.example.com -dataAddress
mysys3.example.com mysys4.example.com
Enter password:
Setup ssh configuration on local system...............................................OK
Setup ssh configuration on mysys1.example.com..........................................OK
Setup ssh configuration on mysys2.example.com..........................................OK
Setup ssh configuration on mysys3.example.com..........................................OK
Setup ssh configuration on mysys4.example.com..........................................OK
Setup passwordless ssh from local system to mysys1.example.com.........................OK
Setup passwordless ssh from local system to mysys2.example.com.........................OK
Setup passwordless ssh from local system to mysys3.example.com.........................OK
Setup passwordless ssh from local system to mysys4.example.com.........................OK
```

```
Setup passwordless ssh from mysys1.example.com to mysys1.example.com.....................OK
Setup passwordless ssh from mysys1.example.com to mysys2.example.com.....................OK
Setup passwordless ssh from mysys1.example.com to mysys3.example.com.....................OK
Setup passwordless ssh from mysys1.example.com to mysys4.example.com.....................OK
Setup passwordless ssh from mysys2.example.com to mysys1.example.com.....................OK
Setup passwordless ssh from mysys2.example.com to mysys2.example.com.....................OK
Setup passwordless ssh from mysys2.example.com to mysys3.example.com.....................OK
Setup passwordless ssh from mysys2.example.com to mysys4.example.com.....................OK

Passwordless ssh working between hosts:

From\To           mysys1.example.com mysys2.example.com mysys3.example.com mysys4.example.com
---------         ------------------ ------------------ ------------------ ------------------
*us*              Yes                Yes                Yes                Yes
mysys1.example.com Yes               Yes                Yes                Yes
mysys2.example.com Yes               Yes                Yes                Yes
mysys3.example.com N/A               N/A                N/A                N/A
mysys4.example.com N/A               N/A                N/A                N/A
```

Notes

- In specifying host addresses, for each host use the same format—fully qualified domain name, simple host name, or IP address—that is used in the -internalAddress or -address option of the hostCreate (or gridCreate) command. For example, do not specify mysys1 for gridSshConfig then mysys1.example.com for hostCreate.

- You can run gridSshConfig multiple times without harm. If you want to enable passwordless SSH on additional hosts later, you can run the command again for those hosts without impacting the hosts already configured.

- In the event of any failure during execution, the command will continue to complete the configuration on as many hosts as it can.

- "Permission denied" errors in the error logs may indicate the password you provided was incorrect or that there is another permissions issue that prevents the command from completing successfully (for example, inappropriate permissions for the user home directory, where the .ssh directory is placed).

Host operations

Use ttGridAdmin commands in this section to define a host in the model, modify a host, delete a host, execute commands on all hosts, or list all hosts.

Create a host (hostCreate)

The hostCreate command defines a host in the model.

```
ttGridAdmin hostCreate [name]
                       [-address addr]
                       [-internalAddress addr]
                       [-externalAddress addr]
                       [-dataspacegroup n]
                       [-nodataspacegroup]
                       [-physicalgroup group1 [group2 [group3 [...]]]]
                       [-nophysicalgroup]
                       [-like name [-cascade]]
                       [-comment comment]
```

Options

The hostCreate command has the options:

Option	Description
name	Specifies the name for the host object in the model. The default is the first component of the operating system host name (the host name up to but not including the first ".", such as myhost).
	If this option is omitted, the host system must be accessible through passwordless SSH at the time hostCreate is executed.
-internalAddress addr	DNS name or IP address of the host for internal communications, inside the grid. Use this together with -externalAddress. The host must be accessible by passwordless SSH at the specified address.
	This option takes one name or address only, and a specified name must resolve to one IP address or to multiple IP addresses on the same network segment.
	If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid.
	Also see Notes below and "Address formats" on page 4-6.
-externalAddress addr	DNS name or IP address of the host for external communications, outside the grid, for client/server connections. Use this together with -internalAddress. The host must be accessible by passwordless SSH at the specified address.
	This option takes one name or address only, but a name may resolve to one or more IP addresses.
	If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid.
	Also see Notes below and "Address formats" on page 4-6.

Option	Description
-address *addr*	DNS name or IP address of the host for both external and internal communications, if a single address is used. The host must be accessible by passwordless SSH at the specified address.
	This option takes one name or address only, and a specified name must resolve to one IP address or to multiple IP addresses on the same network segment.
	If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid.
	Note: Using a single address is not recommended for production environments.
	Also see Notes below and "Address formats" on page 4-6.
-dataspacegroup *n*	Specifies that this host will belong to data space group number *n*. The number of data space groups a grid has is determined by the k value set for the grid.
	A host with a data instance must belong to a data space group.
	Also see -nodataspacegroup and Notes below.
	Note: Once a host is assigned to a data space group and modelApply is executed, you cannot change the assignment.
-nodataspacegroup	Specifies that the host will not be assigned to a data space group. This is the default.
	A host with a data instance must belong to a data space group.
	Also see -dataspacegroup and Notes below.
-physicalgroup *group1* [*group2* [*group3* [...]]]	Specifies the set of physical groups that this host will be associated with.
	It is advisable to spread data instances between different physical groups so that there is redundancy in case of failure. See "Assigning hosts to data space groups" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.
	Also see -nophysicalgroup.
	Note: Physical group assignments are considered by the dataSpaceGroupSuggest command. See Notes below.
-nophysicalgroup	Specifies that the host will be associated with no physical groups. This is the default.
	Also see -physicalgroup.
-like *name*	Specifies that this new host should be created with the same attributes as the named existing host, except where other options that you specify override settings from the existing host.
	Also see -cascade.
-cascade	Use this option with the -like option to specify that installations and instances associated with the -like host also be defined for the host being created. (These objects will be defined for the new host, but not actually created until you run modelApply.)
-comment *comment*	Associates a comment with the host object. Put the comment in quotes if there are any spaces. The comment is stored and included in output of the hostList command.

Examples

Create a second management instance by adding a new host to the model with a set of installations and instances identical to those on the existing host (specified in the -like

option). This command is run from the first management instance, which is on the first host, `mysys1host` (defined earlier, in the example for "Create a grid (gridCreate)" on page 4-56):

```
% ttGridAdmin hostCreate mysys2host -internalAddress intmysys2.example.com
-externalAddress extmysys2.example.com -like mysys1host -cascade
Host mysys2host created in Model
Installation installation1 created in Model
Instance gridmgmt1 created in Model
```

This defines `gridmgmt1` on `mysys2host`, duplicating `gridmgmt1` on `mysys1host`.

Create a host for a data instance, specifying the data space group:

```
% ttGridAdmin hostCreate mysyshost3 -internalAddress intmysys3.example.com
-externalAddress extmysys3.example.com -dataSpaceGroup 1
Host mysyshost3 created in Model
```

Notes

- In specifying host addresses, for each host use the same format—fully qualified domain name, simple host name, or IP address—as was used in the `gridSshConfig` command for that host. For example, do not specify `mysys1` for `hostCreate` if `mysys1.example.com` was specified for `gridSshConfig`.

- You can use `hostModify` to change some settings later.

- If you do not assign the host to a data space group during host creation, you can later use the `dataSpaceGroupSuggest` command to determine optimal assignments based on which physical groups are associated with each host, or you can use the `hostModify` command to choose a data space group manually.

- Hosts on the grid may be configured with either one or two network addresses, depending on system topology. If configured with two addresses, one is used for communications with systems inside the grid (internal) and one is used for client/server access to databases inside the grid from systems outside the grid (external). If configured with one address, which is not recommended for production environments, it is used for both internal and external communications. You must either set `-address` or set `-internalAddress` and `-externalAddress`.

Delete a host (hostDelete)

The `hostDelete` command removes a host from the model.

```
ttGridAdmin hostDelete name
                        [-cascade]
```

Options

The `hostDelete` command has the options:

Option	Description
name	Name of the host object to remove from the model.
-cascade	Specifies that installation and instance objects associated with the host should also be removed from the model.

Examples

This deletes a host that was created in an example in "Create a host (hostCreate)" on page 4-65.

```
% ttGridAdmin hostDelete mysys2host -cascade
Instance gridmgmt1 on Host mysys2host deleted from Model
Installation installation1 on Host mysys2host deleted from Model
Host mysys2host deleted from Model
```

Notes

- If the host has any installations or instances, you must either use -cascade or use installationDelete and instanceDelete.

- Deleting instances and installations removes the objects from the model but does not remove the physical instances and installations.

Execute a command or script on grid hosts (hostExec)

The hostExec command executes a command (such as a system command or TimesTen command) or a script on hosts in the grid, as specified.

```
ttGridAdmin hostExec [-only hostname]
                     [-exclude hostname]
                     [-parallel n]
                      command | -script filepath
```

Options

The hostExec command has the options:

Option	Description
-only *hostname*	The command or script is executed only on the specified hosts. Specify just one host with -only, but you can use -only multiple times on the command line.
	Use host names as defined in the model.
	Without -only or -exclude, the command or script is executed on all hosts in the model.
-exclude *hostname*	The command or script is executed on all hosts in the grid except for the specified hosts. Specify just one host with -exclude, but you can use -exclude multiple times on the command line.
	Use host names as defined in the model.
	Without -only or -exclude, the command or script is executed on all hosts in the model.
-parallel *n*	Specifies that the command or script executes on no more than *n* hosts simultaneously. The default is 10. A value of 1 results in serial execution.
command | -script *filepath*	*command* specifies a command to run.
	Or:
	-script *filepath* specifies the path and name of a shell script to run. The script must be on the local system, then is copied to each host.

Examples

This example first shows the existing hosts in the grid, then uses hostExec to run the df / command (to show disk space) on each host, excluding mysys3host and mysys4host. So the command is executed on mysys1host and mysys2host.

```
% ttGridAdmin hostList
Name        IntAddress            ExtAddress            DSG Comment
----------- --------------------- --------------------- --- -------
mysys1host  intmysys1.example.com extmysys1.example.com  1
mysys2host  intmysys2.example.com extmysys2.example.com  2
mysys3host  intmysys3.example.com extmysys3.example.com  1
mysys4host  intmysys4.example.com extmysys4.example.com  2

% ttGridAdmin hostExec -exclude mysys3host -exclude mysys4host df /
Commands executed on:
  mysys1host rc 0
  mysys2host rc 0
Return code from mysys1host: 0
Output from mysys1host:
Filesystem          1K-blocks     Used Available Use% Mounted on
/dev/xvda2          173483816 28416336 136254988  18% /
Return code from mysys2host: 0
Output from mysys2host:
Filesystem          1K-blocks     Used Available Use% Mounted on
/dev/xvda2          117144964 35319512  75874836  32% /
```

Notes

- The command or script is executed on each host as the instance administrator, through passwordless SSH.

- No environment variables are set on the hosts, other than those set by SSH by default.

- The command returns 2000 if execution did not complete prior to the timeout.

- During execution, stdout and stderr output is displayed as part of the stdout and stderr output from the hostExec command. Because output is buffered, the output from different commands is not intermingled.

List all hosts in the model (hostList)

The hostList command lists information about hosts in the specified version of the model.

```
ttGridAdmin hostList [-latest|-current|-version n]
```

Options

The hostList command has the options:

Option	Description
-latest	Lists hosts in the latest model—the model being modified and not yet applied to the grid. This is the default.
-current	Lists hosts in the current model—the model most recently applied to the grid.
-version n	Lists hosts in the specified version number of the model.

Examples

The following two examples, relating to examples shown in "Modify a host (hostModify)" on page 4-70 and "List model versions (modelList)" on page 4-104, show identical output, indicating that version 4 is the latest version (the version not yet applied to the model).

For each host, the host name, internal address, external address, and associated data space group are listed (optionally with a comment).

```
% ttGridAdmin hostlist
Name        IntAddress            ExtAddress           DSG Comment
----------- --------------------- ------------------   --- --------------------
mysys1host  intmysys1.example.com extmysys1.example.com  1
mysys2host  intmysys2.example.com extmysys2.example.com  1
mysys3host  intmysys3.example.com extmysys3.example.com  1 Move from location1.
mysys4host  intmysys4.example.com extmysys4.example.com  2

% ttGridAdmin hostlist -version 4
Name        IntAddress            ExtAddress           DSG Comment
----------- --------------------- ------------------   --- --------------------
mysys1host  intmysys1.example.com extmysys1.example.com  1
mysys2host  intmysys2.example.com extmysys2.example.com  1
mysys3host  intmysys3.example.com extmysys3.example.com  1 Move from location1.
mysys4host  intmysys4.example.com extmysys4.example.com  2
```

Modify a host (hostModify)

The `ttGridAdmin hostModify` command modifies a host object in the model.

```
ttGridAdmin hostModify name
                        [-physicalgroup group1 [group2 [group3 [...]]]]
                        [-addphysicalgroup group1 [group2 [group3 [...]]]]
                        [-removephysicalgroup group1 [group2 [group3 [...]]]]
                        [-nophysicalgroup]
                        [-dataspacegroup n]
                        [-nodataspacegroup]
                        [-comment comment]
```

Options

The `hostModify` command has the options:

Option	Description
name	Name of the existing host object to modify.
-physicalgroup *group1* [*group2* [*group3* [...]]]	Specifies a new set of physical groups that this host will be associated with. All physical groups previously associated with the host will be replaced with the specified groups.
	Also see -addphysicalgroup, -removephysicalgroup, and -nophysicalgroup.
-addphysicalgroup *group1* [*group2* [*group3* [...]]]	Adds the specified physical groups to the groups the host is associated with.
	Also see -physicalgroup, -removephysicalgroup, and -nophysicalgroup.
-removephysicalgroup *group1* [*group2* [*group3* [...]]]	Removes the specified physical groups from the groups the host is associated with.
	Also see -physicalgroup, -addphysicalgroup, and -nophysicalgroup.

Option	Description
-nophysicalgroup	Specifies that the host will not be associated with any physical groups, removing any prior associations.
	Also see -physicalgroup, -addphysicalgroup, and -removephysicalgroup.
-dataspacegroup *n*	Specifies the number of the data space group that this host will belong to. The number of data space groups a grid will have is determined by the k value set for the grid.
	A host with a data instance should always belong to a data space group.
-nodataspacegroup	Specifies that this host will not be part of any data space group (the default)
-comment *comment*	Associates a comment with the host object or modifies an existing comment. Put the comment in quotes if there are any spaces. The comment is stored and included in output of the hostList command.

Examples

```
% ttGridAdmin hostModify mysyshost2 -physicalGroup location3
-comment "Move from location1."
Host mysyshost2 modified in Model
```

Notes

- The host system must be accessible through passwordless SSH at the time hostModify is executed.

- If modelApply has already been executed for a model including this host, you cannot change the data space group assignment.

Import and export operations

Use `ttGridAdmin` commands in this section to import and export databases, display the status of those operations, or delete an export.

Also see "Migrating, Backing Up and Restoring Data" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Export a database (dbExport)

The `dbExport` command exports data from the specified database into a specified repository. The `dbExport` and `dbImport` commands are used, for example, to migrate a database between two grids or between versions of TimesTen that are not patch-compatible. See "Migrating, Backing Up and Restoring Data" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for additional information.

```
ttGridAdmin dbExport dbname
                    -repository reponame
                    [-name exportname]
```

An export is stored as a *collection* under a *repository*. You first must create the repository. See

Options

The `dbExport` command has the options:

Option	Description
dbname	Name of the database to export.
-repository *reponame*	Name of the repository where the export will be stored.
-name *exportname*	Specifies a name for the export. The default is the letter "M" followed by the date and time of the backup, in the format: *Myyyymmddhhmmss*

Examples

```
% ttGridAdmin dbExport database1 -repository repo1 -name exp_db1
...
dbExport exp_db1 started
```

You can then use `dbExportStatus` to check progress, as shown in the example in The export is finished when each element and the database as a whole are indicated as complete.

Notes

- The export is performed asynchronously. Use the `dbExportStatus` command to check progress.

- Each replica set of the database is stored as a sub-collection.

- The database must be in a closed state with all connections closed when you run `dbExport`.

- Only one `dbExport` command can be run for a database at any given time, and `dbExport` cannot run concurrently with `dbImport`.

■ For disk space requirements, see "Exporting and importing a database" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Delete a database export (dbExportDelete)

The dbExportDelete command deletes the specified database export.

```
ttGridAdmin dbExportDelete -repository reponame
                              -name exportname
```

Options

The dbExportDelete command has the options:

Option	Description
-repository *reponame*	Name of the repository where the export is stored.
-name *exportname*	Name of the export to delete.

Examples

This example deletes the export created in "Export a database (dbExport)" on page 4-72.

```
% ttGridAdmin dbExportDelete -repository repo1 -name exp_db1
Export exp_db1 deleted
```

Notes

This command is typically used to delete old or failed exports.

Display the status of a database export (dbExportStatus)

The dbExportStatus command shows the status of a database export or exports previously started.

```
ttGridAdmin dbExportStatus dbname
                              [-name exportname]
```

Options

The dbExportStatus command has the options:

Option	Description
dbname	Name of the database being exported.
-name *exportname*	Name of the export to check. The default is all exports of the specified database.

Examples

This example shows status upon completion of the export from the example in "Export a database (dbExport)" on page 4-72. (That is the only export for database1 in the repository.)

```
% ttGridAdmin dbExportStatus database1
Database  Export  Repository Host  Instance  Elem State     Started
--------- ------- ---------- ----- --------- ---- --------- ------------------------
database1 exp_db1 repo1                            Completed 2017-03-02T14:42:24.000Z
                             host3 instance1     1 Complete
```

```
                host4 instance1   2 Complete
                host5 instance1   3 Complete
```

Notes

When you believe the export is complete, confirm that dbExportStatus shows Complete for the export as a whole and for every instance. If there were any failures, see "Check the status of a database export" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Import a database (dbImport)

The dbImport command imports data from a specified previous export into the specified database. The dbExport and dbImport commands are used, for example, to migrate a database between two grids or between versions of TimesTen that are not patch-compatible.

```
ttGridAdmin dbImport dbname
                     -repository reponame
                     -name exportname
                     [-ckptFreq mb]
                     [-updateStats]
                     [-estimateStats pct]
                     [-numThreads num]
```

Options

The dbImport command has the options:

Option	Description
dbname	Name of the database where the data is to be imported.
-repository *reponame*	Name of the repository where the export is located.
-name *exportname*	Name of the export to use for the import.
-ckptFreq *mb*	Checkpoint frequency, in terms of how many megabytes have been imported. A checkpoint is written each time that many megabytes have been imported. The default is to write no checkpoints during the import.
-updateStats	Update statistics on each table as it is imported. Also see Notes below.
-estimateStats *pct*	Estimate statistics on each table as it is imported, by reading the specified percentage of rows of each table. Also see Notes below.
-numThreads *num*	Restore database objects in parallel using the specified number of threads. Valid values are 1 through 32. The default value is 4.

Examples

This example imports the export created in the example in "Export a database (dbExport)" on page 4-72, into a database imp_db1.

```
% ttGridAdmin dbImport imp_db1 -repository repo1 -name exp_db1
dbImport exp_db1 started
```

You can then use dbImportStatus to check progress, as shown in the example in "Display the status of a database import (dbImportStatus)" on page 4-75. The import is finished when each element and the database as a whole are indicated as complete.

Notes

- The database must already be created and loaded and must have a distribution map, but must be closed, with all connections closed, when you run dbImport.

- If you specify both -estimateStats and -updateStats, statistics on imported tables are updated, not estimated.

- The import is performed asynchronously. Use the dbImportStatus command to check progress.

- Only one dbImport command can run for a database at any given time, and dbImport cannot run concurrently with dbExport.

- Functionality of the -ckptFreq, -updateStats, and -estimateStats options is the same as for equivalent options of the ttMigrate utility. See "ttMigrate" on page 5-98.

- For disk space requirements, see "Exporting and importing a database" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Display the status of a database import (dbImportStatus)

The dbImportStatus command shows the status of a database import previously started.

```
ttGridAdmin dbImportStatus dbname
                           [-name exportname]
```

Options

The dbImportStatus command has the options:

Option	Description
dbname	Name of the database where the import is being checked.
-name *exportname*	Name of the export from which the data is being imported. You can use this option in the atypical scenario where there are multiple imports into the same database (otherwise, the status of all the imports would be shown).

Examples

This example shows status upon completion of the import from the example in "Import a database (dbImport)" on page 4-74.

```
% ttGridAdmin dbImportStatus imp_db1 -name exp_db1
Database Import  Repository Host  Instance  Elem State                  Started
-------- ------- ---------- ----- --------- ---- -------------------- -----------------------
imp_db1  exp_db1 repo1                            Import_Finale_Complete 2016-07-25T17:53:27.000Z
                            host1 instance1    1 Import_Rows_Complete
                            host3 instance1    3 Import_Rows_Complete
```

Notes

When you believe the import is complete, confirm that dbImportStatus shows Complete for the import as a whole and for every instance. If there were any failures,

see "Check the status of a database import" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Installation operations

Use `ttGridAdmin` commands in this section to define a TimesTen installation in the model, list all installations in the grid, show status of all installations, delete an installation, or execute a command on all installations.

Create an installation (installationCreate)

The `installationCreate` command defines a TimesTen installation in the model.

```
ttGridAdmin installationCreate hostname[.installname]
                              -location path
                              [-source where]
                              [-comment comment]
```

Options

The `installationCreate` command has the options:

Option	Description
hostname[.installname]	The *hostname* is the name of the host where the installation is to be created, optionally with a specified *installname* for the name of the installation in the model. The default is `installation1`.
-location path	Path, on the specified host, to the directory where the installation is to be created. The specified directory does not have to exist, but if it exists it must be empty.
-source where	Location that the installation will be copied from. The location does not have to be on a system that is part of the grid. You can specify it in any of the following formats, as applicable:
	/path *address*:/path *address*!/path [*address*]:/path
	If *path* is a directory, it must be the top-level `tt18.1.4.1.0` directory of an existing TimesTen installation. If it is a file, it must be a `.zip` file that expands into a TimesTen installation. The *address* is a DNS name or IP address.
	If *address* is specified, passwordless SSH is used to fetch the installation source from the system with that address. You must use the fourth format if there is a colon in the address itself, such as for IPv6 addresses.
	The default is the location of the installation associated with the active management instance, from which `ttGridAdmin` is executed.
	Also see "Address formats" on page 4-6.
-comment comment	Associates a comment with the installation object. Put the comment in quotes if there are any spaces. The comment is stored and included in output of the `installationList` command.

Examples

Create an installation for host `mysys4host`, using the default source location. (This example was run from `mysys1`.)

```
% ttGridAdmin installationCreate mysys4host.installcreate
```

```
-location /sw/tten/grid/ttinstls/installcreate
Installation installcreate on Host mysys4host created in Model
```

This time, specify a source location:

```
% ttGridAdmin installationCreate mysys4host.installcreate2
-location /sw/tten/grid/ttinstls/installcreate2 -source
mysys1:/sw/tten/grid/ttinstls/myinstl/tt18.1.4.1.0
Installation installcreate2 on Host mysys4host created in Model
```

Notes

- This command does not create a physical installation. It defines an installation object in the model. (The `modelApply` command creates the installation.)

- Multiple installation objects for the same TimesTen release can point to the same physical installation; however, you cannot specify the same location on the same host for installations from different releases.

Delete an installation (installationDelete)

The `installationDelete` command deletes an installation from the model. It deletes the specified installation (or the only installation, as applicable) on the specified host.

```
ttGridAdmin installationDelete hostname[.installname]
```

Options

The `installationDelete` command has the option:

Option	Description
hostname[.*installname*]	The *hostname* is the name of the host where the installation is to be deleted. The *installname* is the name of the installation to be deleted and is required only if there is more than one installation on the host.

Examples

In this example, `installcreate2` is the only installation on the host.

```
% ttGridAdmin installationDelete mysys4host
Installation installcreate2 on Host mysys4host deleted from Model
```

Notes

- You cannot remove an installation that is still used by instances on the specified host.

- This command removes the installation object from the model but does not remove the physical installation. Remove the files manually when you are certain they are no longer used.

Execute a command or script on grid installations (installationExec)

The `installationExec` command executes a command (such as a system command or TimesTen command) or a script on installations in the grid, as specified.

```
ttGridAdmin installationExec [-only hostname[.installname]]
                             [-exclude hostname[.installname]]
                             [-parallel n]
                             command | -script filepath
```

As the command or script executes, the `TIMESTEN_INSTALL` environment variable is set to contain the fully qualified path name of the installation as defined in the model.

Options

The `installationExec` command has the options:

Option	Description
-only *hostname*[.*installname*]	The command or script is executed only on the specified installations. Specify just one installation with -only, but you can use -only multiple times on the command line.
	Use host names and installation names as defined in the model. You do not have to include the installation name if it is the only installation on the host.
-exclude *hostname*[.*installname*]	The command or script is executed on all installations in the grid except for those specified. Specify just one installation with -exclude, but you can use -exclude multiple times on the command line.
	Use host names and installation names as defined in the model. You do not have to include the installation name if it is the only installation on the host.
-parallel *n*	Specifies that the command or script executes on no more than *n* installations simultaneously. The default is 10. A value of 1 results in serial execution.
command | -script *filepath*	*command* specifies a command to run.
	Or:
	-script *filepath* specifies the path and name of a shell script to run. The script must be on the local system, then is copied to each installation.

Examples

This example checks the disk space usage on the file system of each installation.

```
% ttGridAdmin installationExec df '$TIMESTEN_INSTALL'
Commands executed on:
  mysys2host.installation1 rc 0
  mysys4host.installadc rc 0
  mysys1host.installation1 rc 0
  mysys3host.installslc rc 0
Return code from mysys2host.installation1: 0
Output from mysys2host.installation1:
Filesystem          1K-blocks     Used Available Use% Mounted on
/dev/xvda2          117144964 42660228  68534120  39% /
Return code from mysys4host.installadc: 0
Output from mysys4host.installadc:
Filesystem          1K-blocks     Used Available Use% Mounted on
/dev/xvda2          117144964 42660228  68534120  39% /
Return code from mysys1host1.installation1: 0
Output from mysys1host.installation1:
Filesystem          1K-blocks     Used Available Use% Mounted on
/dev/xvda2          173483816 57971304 106700020  36% /
Return code from mysys3host.installslc: 0
Output from mysys3host.installslc:
Filesystem          1K-blocks     Used Available Use% Mounted on
/dev/xvda2          173483816 57971312 106700012  36% /
```

Notes

- The command or script is executed as the instance administrator on each installation, through passwordless SSH.

- The command returns 2000 if execution did not complete prior to the timeout.

- During execution, stdout and stderr output is displayed as part of the stdout and stderr output from the installationExec command. Because output is buffered, the output from different commands is not intermingled.

List installations (installationList)

The installationList command lists all TimesTen installations in the model.

```
ttGridAdmin installationList [-latest|-current|-version n]
                             [-instance]
```

Options

The installationList command has the options:

Option	Description
-latest	List the installations in the latest (in progress) model, which has not yet been applied to the grid. This is the default.
-current	List the installations in the current model—the model currently applied to the grid.
-version n	List the installations in the specified version number of the model.
-instance	Show the instances that are using each installation. Installations not yet associated with an instance are not displayed.

Examples

This example lists the installations in the latest (in progress) model.

```
% ttGridAdmin installationList
Host         Install        Location                                    Comment
-----------  -------------  ------------------------------------------  -------
mysys1host   installation1  /sw/tten/grid/ttinstls/myinstl/tt18.1.4.1.0/
mysys2host   installation1  /sw/tten/grid/ttinstls/myinstl/tt18.1.4.1.0/
mysys3host   installcreate1 /sw/tten/grid/ttinstls/installcreate1/
mysys4host   installcreate1 /sw/tten/grid/ttinstls/installcreate1/
```

This example lists installations in the latest model that are associated with an instance.

```
% ttGridAdmin installationList -instance
Host         Install        Instance  Location                                    Comment
-----------  -------------  --------  ------------------------------------------  -------
mysys1host   installation1  gridmgmt1 /sw/tten/grid/ttinstls/myinstl/tt18.1.4.1.0/
mysys2host   installation1  gridmgmt1 /sw/tten/grid/ttinstls/myinstl/tt18.1.4.1.0/
```

Display status of installations (installationStatus)

The installationStatus command shows the status of all installations that are associated with the grid. This is status of the physical installations, not status of installations in the model.

```
ttGridAdmin installationStatus
```

Examples

```
% ttGridAdmin installationStatus
Host        Install          Usable DelPend Message When
----------- ---------------- ----   ------- ------- ------------------
mysys1host  installation1    Yes    N               2016-11-01 14:49:31
mysys2host  installation1    Yes    N               2016-11-01 14:49:31
mysys3host  installcreate2slc Yes   N               2016-11-01 14:49:31
mysys4host  installcreate2adc Yes   N               2016-11-01 14:49:31
```

The `DelPend` entry indicates whether a deletion is pending, where `installationDelete` was executed but the updated model has not yet been applied to remove the physical installation.

Instance operations

Use `ttGridAdmin` commands in this section to define a TimesTen Scaleout instance in the model, modify an instance, delete an instance, list instances in the grid, display status of instances in the grid, import or export an instance configuration file, or execute a command on instances in the grid.

Export instance configuration attributes (instanceConfigExport)

The `instanceConfigExport` command exports configuration attribute settings, previously imported using `instanceConfigImport`, from the specified version of the model.

```
ttGridAdmin instanceConfigExport [-latest|-current|-version n]
                                 [filepath]
```

Options

The `instanceConfigExport` command has the options:

Option	Description
-latest	Export configuration attribute settings that were imported into the latest model, which has not yet been applied to the grid. This is the default.
-current	Export configuration attribute settings that were imported into the current model—the model currently applied to the grid.
-version n	Export configuration attribute settings that were imported into the specified version of the model.
filepath	The path and name of the file to export configuration attribute settings into. If no file is specified, the information is written to stdout.

Examples

This example exports a configuration attribute setting from the current version of the model and from the latest (default) version of the model after the imports shown in the next section, "Import instance configuration attributes (instanceConfigImport)" on page 4-82. Contents of the export files are also shown.

```
% ttGridAdmin instanceConfigExport -current /tmp/instanceconfigexp1
% more /tmp/instanceconfigexp1
max_conns_per_server=500
% ttGridAdmin instanceConfigExport /tmp/instanceconfigexp2
% more /tmp/instanceconfigexp2
max_conns_per_server=1000
```

Notes

This command exports only settings that were previously imported, not any other settings from the `timesten.conf` files.

Import instance configuration attributes (instanceConfigImport)

The `instanceConfigImport` command imports configuration attribute settings into the latest version of the model, to be used by every instance in the grid.

```
ttGridAdmin instanceConfigImport [filepath]
```

After you execute `modelApply`, the configuration file for each instance is updated to include the imported attributes. You must restart the TimesTen daemon on each instance for the changes to take effect.

See Notes below for a list of attributes you cannot import.

Options

The `instanceConfigImport` command has the option:

Option	Description
filepath	The path and name of the file to import configuration attribute settings from. If no file is specified, the information is read from `stdin`.

Examples

Import from this file.

```
% more /tmp/instanceconfigimp1
# Set maximum number of connections.
max_conns_per_server=500
% ttGridAdmin instanceConfigImport /tmp/instanceconfigimp1
Instance configuration file /tmp/instanceconfigimp1 imported
```

Apply the model (output is not shown):

```
% ttGridAdmin modelApply
...
ttGridAdmin modelApply complete
```

Now import from this file:

```
% more /tmp/instanceconfigimp2
# Set maximum number of connections.
max_conns_per_server=1000
% ttGridAdmin instanceConfigImport /tmp/instanceconfigimp2
Instance configuration file /tmp/instanceconfigimp2 imported
```

After these steps, the latest version of the model will have a maximum connections setting of 500 and the current version of the model will have a setting of 1000. This is shown in the examples in the previous section, "Export instance configuration attributes (instanceConfigExport)" on page 4-82.

Notes

- As shown in the example, each entry that is imported is of the form *name=value*. You can also include comments, indicated by `#`.

- The `timesten.conf` files are updated when you execute `modelApply`.

- The following attributes are set automatically when the `modelApply` command creates or configures instances and cannot be imported:

```
admin_uid
admin_user
client_only
daemon_port
grid_external_addr
grid_guid
grid_host
```

```
grid_instance
grid_internal_addr
grid_name
guid
hostname
instance_guid
instance_name
listen_addr
server_port
timesten_release
tns_admin
```

- Refer to Chapter 1, "TimesTen Instance Configuration File" for information about TimesTen configuration attributes.

Create an instance (instanceCreate)

The instanceCreate command defines an instance in the model.

```
ttGridAdmin instanceCreate   hostname[.instancename]
                             -location path
                             [-type management|data]
                             [-installation name]
                             [-daemonport n]
                             [-csport n]
                             [-mgmtport n]
                             [-comment comment]
                             [-walletDir path]
```

Options

The instanceCreate command has the options:

Option	Description
hostname[.*instancename*]	The *hostname* is the name of the host where the instance is to be created, optionally with a specified *instancename* for the name of the instance in the model. The default is instance1.
-location *path*	Path, on the specified host, to the directory where the instance is to be created. The specified directory does not have to exist.
-type management|data	Specifies which type of instance is defined. The default is a data instance.
-installation *name*	Name of the installation that the instance will use. This option is not necessary if there is only one installation on the host.
-daemonport *n*	Port number where the TimesTen main daemon for the instance will listen. The default is 6624. **Important**: If you create more than one instance on a system (such as a management instance and a data instance), you must specify unique port numbers.
-csport *n*	Port number where the server for TimesTen client/server will listen. The default is 6625. **Important**: If you create more than one instance on a system (such as a management instance and a data instance), you must specify unique port numbers.
-mgmtport *n*	For management instances, the port number that will be used by replication when management data on the active management instance is replicated. The default is 3754.

Option	Description
-comment *comment*	Associates a comment with the instance object. Put the comment in quotes if there are any spaces. The comment is stored and included in output of the instanceList command.
-walletDir *path*	For the instance being crated, path to the directory where the Oracle Wallet with cryptographic information for this instance will be stored. The default is *timesten_home*/info.
	Wallets for multiple instances can be stored in the same directory, which can be shared between the instances, such as through NFS.

Examples

```
% ttGridAdmin instanceCreate mysys3host.griddata1
-location /sw/tten/grid/ttinstances -daemonPort 20000 -csPort 21000
Instance griddata1 on Host mysys3host created in Model
```

Notes

- This command does not create a physical instance. It defines an instance object in the model. The modelApply command creates the physical instance.

- Be aware of these prerequisites:

  - The host must have an associated installation object. Use the installationCreate command.

  - For a data instance, the host must be in a data space group. If that is not the case, the physical instance cannot be created when you apply the model.

    You can use the hostList command to confirm whether a host is in a data space group, and the hostModify command to assign a data space group if needed.

- The *timesten_home* directory will be *location/name*. In the example, where the location is /sw/tten/ttinstances and the instance name is griddata1, *timesten_home* will be /sw/tten/ttinstances/griddata1.

- Some instance settings can be changed later through the instanceModify command, as desired.

Delete an instance (instanceDelete)

The instanceDelete command deletes an instance from the model.

```
ttGridAdmin instanceDelete hostname[.instancename]
```

Options

The instanceDelete command has the option:

Option	Description
hostname[.*instancename*]	The *hostname* is the name of the host where the instance is to be deleted. The *instancename* is the name of the instance to be deleted and is required only if there is more than one instance on the host.

Examples

In this example, griddata1 is the only instance on the host.

```
% ttGridAdmin instanceDelete mysys3host
Instance griddata1 on Host mysys3host deleted from Model
```

Notes

- This command first stops the instance if it has not already been stopped.

- The command removes the instance object from the model. It does not remove the physical instance. (The modelApply command removes the instance.)

- You cannot remove an instance that is still used by other objects in the model.

- You cannot remove an instance that contains a database element.

Execute a command or script on grid instances (instanceExec)

The instanceExec command executes a command (such as a system command or TimesTen command) or a script on instances in the grid, as specified.

```
ttGridAdmin instanceExec [-only hostname[.instancename]]
                         [-exclude hostname[.instancename]]
                         [-parallel n]
                         [-type all|management|data]
                         [-up]
                          command | -script filepath
```

Options

The instanceExec command has the options:

Option	Description
-only hostname[.instancename]	The command or script is executed only on the specified instances. Specify just one instance with -only, but you can use -only multiple times on the command line.
	Use host names and instance names as defined in the model. You do not have to include the instance name if it is the only instance on the host.
-exclude hostname[.instancename]	The command or script is executed on all instances in the grid except for those specified. Specify just one instance with -exclude, but you can use -exclude multiple times on the command line.
	Use host names and instance names as defined in the model. You do not have to include the instance name if it is the only instance on the host.
-parallel n	Specifies that the command or script executes on no more than n instances simultaneously. The default is 10. A value of 1 results in serial execution.
-type all|management|data	Specifies whether the command or script is executed on all instances (the default), only management instances, or only data instances.
	This can be used in combination with -only or -exclude.
-up	Specifies that the command or script is executed only on instances that are part of the current membership. The default is to execute commands on all instances (whether they are running or not).

Option	Description
command | -script *filepath*	*command* specifies a command to run.
	Or:
	-script *filepath* specifies the path and name of a shell script to run. The script must be on the local system, then is copied to each instance.

Examples

On each data instance, this example creates directories databases and logs under /data (with no error if the directories already exist).

```
% ttGridAdmin instanceExec -type data mkdir -p /data/{databases,logs}
Overall return code: 0
Commands executed on:
  mysys6host.griddata4 rc 0
  mysys5host.griddata3 rc 0
  mysys3host.griddata1 rc 0
  mysys4host.griddata2 rc 0
Return code from mysys6host.griddata4: 0
Output from mysys6host.griddata4:
Return code from mysys5host.griddata3: 0
Output from mysys5host.griddata3:
Return code from mysys3host.griddata1: 0
Output from mysys3host.griddata1:
Return code from mysys4host.griddata2: 0
Output from mysys4host.griddata2:
```

This example starts the TimesTen daemon on mysys5host.griddata3 (useful, for example, if the element on that instance went down).

```
% ttGridAdmin instanceExec -only mysys5host.griddata3 ttDaemonAdmin -start
Overall return code: 0
Commands executed on:
  mysys5host.griddata3 rc 0
Return code from mysys5host.griddata3: 0
Output from mysys5host.griddata3:
TimesTen Daemon (PID: 7586, port: 6624) startup OK.
```

For each data instance, this example runs the ttIsql monitor command then exits ttIsql. (Only selected portions of the ttIsql connection output and monitoring output are shown.)

```
% ttGridAdmin instanceExec -type data 'ttIsql -e "monitor;quit" -dsn database1'
Overall return code: 0
Commands executed on:
  mysys4host.griddata2 rc 0
  mysys5host.griddata3 rc 0
  mysys6host.griddata4 rc 0
  mysys3host.griddata1 rc 0
Return code from mysys4host.griddata2: 0
Output from mysys4host.griddata2:

Copyright (c) 1996, 2018, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.

connect "DSN=database1";
Connection successful: DSN=database1;...
```

```
      monitor;

        TIME_OF_1ST_CONNECT:          Fri Aug  3 13:47:42 2018
        ...
        PERM_ALLOCATED_SIZE:          262144
        PERM_IN_USE_SIZE:             29997
        PERM_IN_USE_HIGH_WATER:       29997
        TEMP_ALLOCATED_SIZE:          131072
        TEMP_IN_USE_SIZE:             19146
        TEMP_IN_USE_HIGH_WATER:       22352
        ...

      quit;
      Disconnecting...
      Done.
      Return code from mysys5host.griddata3: 0
      Output from mysys5host.griddata3:

      Copyright (c) 1996, 2018, Oracle and/or its affiliates. All rights reserved.
      Type ? or "help" for help, type "exit" to quit ttIsql.

      connect "DSN=database1";
      Connection successful: DSN=database1;...

      monitor;

        TIME_OF_1ST_CONNECT:          Fri Aug  3 13:47:41 2018
        ...
        PERM_ALLOCATED_SIZE:          262144
        PERM_IN_USE_SIZE:             29916
        PERM_IN_USE_HIGH_WATER:       29932
        TEMP_ALLOCATED_SIZE:          131072
        TEMP_IN_USE_SIZE:             19613
        TEMP_IN_USE_HIGH_WATER:       22819
        ...

      quit;
      Disconnecting...
      Done.
      Return code from mysys6host.griddata4: 0
      Output from mysys6host.griddata4:

      Copyright (c) 1996, 2018, Oracle and/or its affiliates. All rights reserved.
      Type ? or "help" for help, type "exit" to quit ttIsql.

      connect "DSN=database1";
      Connection successful: DSN=database1;...

      monitor;

        TIME_OF_1ST_CONNECT:          Fri Aug  3 13:47:41 2018
        ...
        PERM_ALLOCATED_SIZE:          262144
        PERM_IN_USE_SIZE:             29981
        PERM_IN_USE_HIGH_WATER:       29981
        TEMP_ALLOCATED_SIZE:          131072
        TEMP_IN_USE_SIZE:             19344
        TEMP_IN_USE_HIGH_WATER:       22550
        ...
```

```
quit;
Disconnecting...
Done.
Return code from mysys3host.griddata1: 0
Output from mysys3host.griddata1:

Copyright (c) 1996, 2018, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.

connect "DSN=database1";
Connection successful: DSN=database1;...

monitor;

  TIME_OF_1ST_CONNECT:          Fri Aug  3 13:47:40 2018
  ...
  PERM_ALLOCATED_SIZE:          262144
  PERM_IN_USE_SIZE:             29965
  PERM_IN_USE_HIGH_WATER:       29965
  TEMP_ALLOCATED_SIZE:          131072
  TEMP_IN_USE_SIZE:             19281
  TEMP_IN_USE_HIGH_WATER:       22486
  ...

quit;
Disconnecting...
Done.
```

Notes

- The command or script is executed as the instance administrator on each instance, through passwordless SSH.

- Environment variables (such as TIMESTEN_HOME, CLASSPATH, PATH, and LD_LIBRARY_PATH) are set appropriately for each instance.

- The command returns 2000 if execution did not complete prior to the timeout.

- During execution, stdout and stderr output is displayed as part of the stdout and stderr output from the instanceExec command. Because output is buffered, the output from different commands is not intermingled.

List instances (instanceList)

The instanceList command lists information about instances in the specified version of the model.

```
ttGridAdmin instanceList [-latest|-current|-version n]
                         [-type all|management|data]
                         [-install]
```

Options

The instanceList command has the options:

Option	Description
-latest	Lists instances in the latest model—the model being modified and not yet applied to the grid. This is the default.
-current	Lists instances in the current model—the model most recently applied to the grid.

Option	Description
-version *n*	Lists instances in the specified version number of the model.
-type all|management|data	Specifies whether all instances (the default), only management instances, or only data instances are listed.
-install	Shows the installation object associated with each instance.

Examples

This example is for a grid with two hosts on each of two systems. On each system, one host has a management instance and one has a data instance. By default, data instances as well as management instances are listed in the latest model (in the process of being modified and not yet applied).

```
% ttGridAdmin instanceList
Host        Instance  Type Instance Home                       Port  CSPort MgmtPort Comment
----------- --------- ---- ---------------------------------- ----- ------ -------- -------
mysys1host  gridmgmt1 Mgmt /sw/tten/grid/ttinstances/gridmgmt1/ 10000 11000   3754
mysys2host  gridmgmt1 Mgmt /sw/tten/grid/ttinstances/gridmgmt1/ 10000 11000   3754
mysys3host  griddata1 Data /sw/tten/grid/ttinstances/griddata1/ 20000 21000
mysys4host  griddata2 Data /sw/tten/grid/ttinstances/griddata2/ 20000 21000
```

This example also shows the associated installation objects (the Comment column is omitted):

```
% ttGridAdmin -instanceList -install
Host        Instance  Installation  Type  Instance Home                       Port  CSPort MgmtPort
----------- --------- ------------- ----  ---------------------------------- ----- ------ --------
mysys1host  gridmgmt1 installation1 Mgmt  /sw/tten/grid/ttinstances/gridmgmt1/ 10000 11000   3754
mysys2host  gridmgmt1 installation1 Mgmt  /sw/tten/grid/ttinstances/gridmgmt1/ 10000 11000   3754
mysys3host  griddata1 installation1 Data  /sw/tten/grid/ttinstances/griddata1/ 20000 21000
mysys4host  griddata2 installation1 Data  /sw/tten/grid/ttinstances/griddata2/ 20000 21000
```

Modify an instance (instanceModify)

The instanceModify command modifies an existing instance object in the model.

```
ttGridAdmin instanceModify hostname[.instancename]
                           [-installation name]
                           [-mgmtPort n]
                           [-comment comment]
```

Options

The instanceModify command has the options:

Option	Description
hostname[.*instancename*.]	The *hostname* is the name of the host where the instance is to be modified. The *instancename* is the name of the instance to be modified and is required only if there is more than one instance on the host.
-installation *name*	Associates the instance with a different installation on the host, specified by the name of the installation in the model.

Option	Description
-mgmtPort *n*	For management instances, a new port number that will be used for replication when management data on the active management instance is replicated. Changing this is allowed only if there ie exactly one management instance at the time the command is issued, but is relevant only if you plan to have two management instances.
-comment *comment*	Associates a comment with the instance object or modifies an existing comment. Put the comment in quotes if there are any spaces. The comment is stored and included in output of the instanceList command.

Examples

In this example, griddata1 is the only instance on the host.

```
% ttGridAdmin instanceModify mysys3host -installation altinstall -comment
Change_from_installcreate1
Instance griddata1 on Host mysys3host modified in Model
```

(Note that if you have a multi-word comment, you can use underscores instead of spaces to avoid having to put the comment in quotes.)

Notes

- This command is most typically used to patch or upgrade your version of TimesTen by pointing to an installation of the desired release.

- When instanceModify updates are applied by a subsequent modelApply command, the instance is not stopped and reconfigured at that time. Instead, the next time the instance is started, TimesTen Scaleout will detect that the instance configuration does not match the model, and will reconfigure it appropriately.

Display status of instances (instanceStatus)

The instanceStatus command displays information about the status of instances in the grid, in JSON format.

```
ttGridAdmin instanceStatus [-type all|management|data]
```

Options

The instanceStatus command has the option:

Option	Description
-type all|management|data	Specifies whether all instances (the default), only management instances, or only data instances are displayed.

Management instance operations

Use `ttGridAdmin` commands in this section to start, stop, switch, examine, or check status of the management instance or instances. Execute the commands from the appropriate management instance.

> **Note:** Typically, there are two management instances, the active and standby. Before you can perform any grid management functions, a management instance must be started as the active instance, from which you can run `ttGridAdmin`. (Initially, the instance from which you create the grid becomes the active management instance.)
>
> See "Configure your grid" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for details.

See "Managing failover for the management instances" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for related information.

Start the active management instance (mgmtActiveStart)

The `mgmtActiveStart` command starts the current management instance (from which the command is run) as the active management instance.

```
ttGridAdmin mgmtActiveStart
```

Examples

```
% ttGridAdmin mgmtActiveStart
This management instance is now the active
```

Notes

- The current management instance must previously be stopped.

- There cannot be another management instance that has been started as the active instance.

Stop the active management instance (mgmtActiveStop)

The `mgmtActiveStop` command stops the active management instance.

```
ttGridAdmin mgmtActiveStop
```

This command is typically used as the last step in shutting down a grid. Otherwise, if there are two management instances, it is recommended to instead use `mgmtActiveSwitch`.

Examples

```
% ttGridAdmin mgmtActiveStop
Active management instance stopped
```

Notes

- If this command is used in a grid with two management instances (not recommended unless you are shutting down the grid), it can be run from either the active or the standby management instance. Nothing is done automatically to

then promote the standby management instance to active. (See `mgmtActiveSwitch`.)

- If data instances are running, then the database elements currently loaded in them will continue to operate.

- You cannot perform any management operations until you restart the active management instance.

- If data instances have stopped or failed, they cannot be restarted until you restart the active management instance.

Switch the active management instance (mgmtActiveSwitch)

The `mgmtActiveSwitch` command, executed from the current standby management instance, results in that instance becoming the active management instance. The original active management instance is stopped if it can be reached.

```
ttGridAdmin mgmtActiveSwitch [-force]
```

Options

The `mgmtActiveSwitch` command has the option:

Option	Description
-force	Specifies that the command will take effect even if the management instance from which it is run cannot be clearly identified as the standby management instance or is not ideally eligible to become the active management instance.
	Important: Using `-force` will likely result in substantial data loss. Use only as a last resort.

Examples

```
% ttGridAdmin mgmtActiveSwitch
This is now the active management instance
```

Notes

- This command is typically used if the active management instance has failed.

- If or when the original active management instance is back up, you can use `mgmtStandbyStart` to restart it as the standby.

- All data instances in the grid will automatically failover from the previous active management instance to the new active management instance.

Examine management instances (mgmtExamine)

The `mgmtExamine` command examines the management instances and recommends any necessary corrective action. Run the suggested commands.

```
ttGridAdmin mgmtExamine
```

Examples

This example shows output when both management instances are up. Aside from the opening note that they are both up, the output is the same as for the `mgmtStatus` command. See "Display status of management instances (mgmtStatus)" on page 4-95

for descriptions of the columns. (For brevity, the Message column, which had no entries, is not shown in this example.)

```
% ttGridAdmin mgmtExamine
Both active and standby management instances are up. No action required.

Host        Instance  Reachable RepRole(Self) Role(Self) Seq RepAgent RepActive
----------- --------- --------- ------------- ---------- --- -------- ---------
mysys1host  gridmgmt1 Yes       Active        Active     554 Up       Yes
mysys2host  gridmgmt1 Yes       Standby       Standby    554 Up       No
```

This example shows output when the active management instance is down, including recommended actions and commands to run:

```
% ttGridAdmin mgmtExamine
Standby management instance is up, but active is down
Promote the standby to active

Host        Instance  Reachable RepRole(Self) Role(Self) Seq RepAgent RepActive Message
----------- --------- --------- ------------- ---------- --- -------- --------- -------------------
----------------
mysys1host  gridmgmt1 No        Unknown       Unknown        Down     No        Management database
is not available
mysys2host  gridmgmt1 Yes       Standby       Standby    557 Up       No
Recommended commands:
ssh -o StrictHostKeyChecking=yes -o PasswordAuthentication=no -x host1.example.com
/sw/tten/gridsetup/ttinstances/gridmgmt1/bin/ttenv ttGridAdmin mgmtActiveSwitch
```

Notes

One use case is if both management instances fail, and you are not certain which one was the active. Run this command to examine them both and determine which one is "current" or "most recent", then start that one as the active management instance.

Start the standby management instance (mgmtStandbyStart)

The mgmtStandbyStart command starts the current management instance (from which the command is run) as the standby management instance.

```
ttGridAdmin mgmtStandbyStart
```

A typical scenario is when the active management instance fails, you promote the standby to active, then run this command to make the original active management instance become the new standby management instance.

Examples

```
% ttGridAdmin mgmtStandbyStart
Standby management instance started
```

Notes

- The instance must previously be stopped.

- There must be another management instance previously started as the active management instance.

- This command initiates replication between the active and standby management instances, synchronizing management data between them.

Stop the standby management instance (mgmtStandbyStop)

The `mgmtStandbyStop` command stops the standby management instance.

```
ttGridAdmin mgmtStandbyStop
```

Examples

```
% ttGridAdmin mgmtStandbyStop
Standby management instance stopped
```

Notes

- This command can be run from either the active or the standby management instance if they are both operational.

- Usage scenarios include:
  – If the standby has failed
  – If you want to stop the standby for any reason, such as to reboot it or perform maintenance

- The command will fail with an error if there is not an operational standby instance at the time the command is run.

Display status of management instances (mgmtStatus)

The `mgmtStatus` command displays status information for the management instances.

```
ttGridAdmin mgmtStatus
```

(Also see "Examine management instances (mgmtExamine)" on page 4-93. The `mgmtExamine` command recommends actions to repair any reported problems with the management instances.)

Examples

```
% ttGridAdmin mgmtStatus
Host        Instance  Reachable RepRole(Self) Role(Self) Seq RepAgent RepActive
----------- --------- --------- ------------- ---------- --- -------- ---------
mysys1host  gridmgmt1 Yes       Active        Active     554 Up       Yes
mysys2host  gridmgmt1 Yes       Standby       Standby    554 Up       No
```

For each instance displayed:

- **Host** and **Instance** show the name of the instance and the name of the host where it is located.

- **Reachable** indicates whether the command was successful in reaching the instance to determine its state.

- **RepRole(Self)** indicates the recorded role, if any, for the instance in replicating data between management instances.

- **Role(Self)** indicates the recorded role, if any, for the instance.

- **Seq** is the sequence number of the most recent change on the instance. If the `Seq` values are the same, then the two management instances are synchronized; otherwise, the one with the larger `Seq` value has the more recent data.

- **RepAgent** indicates whether the replication agent is running on the instance.

- **RepActive** indicates whether changes by the `mgmtStatus` command to management data on the instance were successful. The `mgmtStatus` command attempts to modify management data on each management instance, but this will not work on the standby management instance, which is read-only.

- **Message** has any further information about the instance. (For brevity, this column is not shown in the example.)

Membership operations

Use `ttGridAdmin` commands in this section to export or import the membership client configuration file. A typical scenario is if you want to make changes to the file.

> **Note:** The membership configuration file is first specified when you create the grid, according to the `gridCreate -membershipConfig` option. See "Create a grid (gridCreate)" on page 4-56.

Export the membership configuration file (membershipConfigExport)

The `membershipConfigExport` command exports the contents of the membership client configuration file from the specified version of the grid model into a specified file.

```
ttGridAdmin membershipConfigExport [-latest|-current|-version n]
                                   [filepath]
```

Options

The `membershipConfigExport` command has the options:

Option	Description
`-latest`	Export the configuration file from the latest model—the model being modified and not yet applied to the grid. This is the default.
`-current`	Export the configuration file from the current model—the model most recently applied to the grid.
`-version n`	Export the configuration file from the specified version number of the model.
`filepath`	The path and name of the file to write the contents of the membership client configuration file to. If no file is specified, the configuration is written to `stdout`.

Examples

```
% ttGridAdmin membershipConfigExport -latest /sw/tten/grid/zkcfg/membership2.conf

% cd /sw/tten/grid/zkcfg
% more membership2.conf
Servers zk1.example.com!2181,zk2.example.com!2181,zk3.example.com!2181
```

The example in the next section will import this file.

Import the membership configuration file (membershipConfigImport)

The `membershipConfigImport` command replaces the membership client configuration file in the latest model of the grid with the specified file.

```
ttGridAdmin membershipConfigImport filepath
```

Also see information for the `gridCreate -membershipConfig` option in "Create a grid (gridCreate)" on page 4-56.

Options

The `membershipConfigImport` command has the option:

Option	Description
filepath	The path and name of the file that contains the new membership configuration.

Examples

This example imports the file created in the example from the preceding section, shown again here:

```
% cd /sw/tten/grid/zkcfg
% more membership2.conf
Servers zk1.example.com!2181,zk2.example.com!2181,zk3.example.com!2181
```

Run the command:

```
% ttGridAdmin membershipConfigImport /sw/tten/grid/zkcfg/membership2.conf
Membership configuration file /sw/tten/grid/zkcfg/membership2.conf imported
```

Notes

- Any membership client configuration changes according to the new file are not applied until you execute the modelApply command.

- Once you execute modelApply, the specified file is copied to each instance of the grid and its settings will take effect on each instance the next time the instance is restarted.

Model operations

Use `ttGridAdmin` commands in this section to apply the latest version of the model to the grid, delete a version of the model, export a version of the model to a JSON file, import a version of the model from a JSON file (to become the latest version), compare two versions of the model, or list information about all versions of the model.

> **Note:** The *latest* version of the model is the version that is pending for edits and updates. It has not yet been applied to the model. The *current* version of the model is the version most recently applied to the model. Only the latest version of the model is editable. All other versions are read-only. When the latest version is applied, it becomes the current version and a copy is made to serve as the initial latest version.

Apply the latest version of the model (modelApply)

The `modelApply` command applies the latest version of the model to the grid, implementing previous operations. This includes actions such as creating physical installations and instances according to installation and instance objects that have been defined in the model.

```
ttGridAdmin modelApply [-nostart]
                       [-details]
```

Options

The `modelApply` command has the options:

Option	Description
-nostart	By default, the `modelApply` command automatically starts new TimesTen Scaleout instances when they are created for the grid. If you specify `-nostart`, the instances are created but not started.
-details	Displays additional information about the operations being performed by the command.

Examples

This example shows typical output.

```
% ttGridAdmin modelApply
Creating new model version............................................OK
Exporting current model (version 1)...................................OK
Identifying any deleted objects.......................................OK
Verifying installations...............................................OK
Creating new installations............................................OK
Verifying instances...................................................OK
Creating new instances................................................OK
Updating grid state...................................................OK
Configuring instance authentication...................................OK
Pushing new configuration files to each instance......................OK
Making model version 1 current, version 2 writable...................OK
Checking ssh connectivity of new instances...........................OK
Starting new management instance......................................OK
Configuring standby management instance..............................OK
ttGridAdmin modelApply complete
```

(Output will vary depending on your situation, such as whether installations or instances in the model already existed, either from being created manually or from any previous `modelApply` commands that were only partially successful.)

Notes

- When a grid is created, version 1 of the model is created automatically. When `modelApply` is executed on the grid for the first time, version 1 of the model is made read-only and version 2 is created. Version 2 is an exact copy of version 1 and is read-write. Version 1 is then applied to the grid. Subsequent changes made to the model are made to version 2, until `modelApply` is executed again, at which time version 3 is created, and so on. There is always a writable version of the model available.

- At any given point, the writable version of the model, which has not yet been applied to the grid, is referred to as the *latest* version. The version that has been applied and is operational in the grid is referred to as the *current* version. (The current version and all previous versions are read-only.)

- The `modelApply` command communicates with each instance in the grid and creates or updates configuration files on each instance, including `timesten.conf`, as needed. The command executes these operations in parallel as much as possible, but still may take a significant amount of time to complete. Complete all the steps in getting from one desired configuration to another desired configuration before applying the model.

- It may not always be possible for `modelApply` to complete all of its operations, such as if a host is down. If there are problems, `modelApply` creates error logs in the `diag` directory of the management instance and indicates the names of those logs. The next time you execute `modelApply`, it will try again to complete any operations that failed previously, in addition to completing any new operations.

- See "Applying the changes made to the model" in *Oracle TimesTen In-Memory Database Scaleout User's Guide* for additional information.

Compare models (modelCompare)

The `modelCompare` command compares two versions of the model and displays a summary of changes between them.

```
ttGridAdmin modelCompare -latest|-current|-version n
                         [-latest|-current|-version m]
```

Options

The `modelCompare` command has the options:

Option	Description
-latest	Specifies that the latest version of the model—the model being modified and not yet applied to the grid—is one of the versions to compare.
	If the command line specifies only one version, that version is compared against the latest version by default.
-current	Specifies that the current version of the model—the model most recently applied to the grid—is one of the versions to compare.
-version *n*	Specifies that model version *n* is one of the versions to compare.

Option	Description
-version *m*	Specifies that model version *m* is one of the versions to compare.

Examples

This example shows that the physical group location4 was added between the current model and the latest model. (Other differences shown are for meta data.)

```
% ttGridAdmin modelCompare -current -latest
6,9c6,8
<   "version"    : 8,
<   "whenCreated" : "2016-12-02T13:13:05.000Z",
<   "applied" : true,
<   "whenApplied" : "2016-12-13T14:57:41.000Z",
---
>   "version"    : 9,
>   "whenCreated" : "2016-12-13T14:57:19.000Z",
>   "applied" : false,
11c10
<   "current" : true,
---
>   "current" : false,
31a31,33
>     },
>     {"type" : "physicalGroup" ,
>      "name" : "location4"
```

Notes

The summary of changes is displayed in UNIX diff format.

Export a version of a model (modelExport)

The modelExport command exports information about the grid for the specified version of the model in JSON format, typically to a specified file.

```
ttGridAdmin modelExport [-latest|-current|-version n]
                        [filepath]
```

Within the grid, the hierarchy of the output includes the following:

```
SQLNet
TNSNames
DataSpaceGroups
Hosts
    PhysicalGroups
    Installations
        Instances
Databases
    Connectables
```

Options

The modelExport command has the options:

Option	Description
-latest	Export the latest model—the model being modified and not yet applied to the grid. This is the default.

Option	Description
-current	Export the current model—the model most recently applied to the grid.
-version *n*	Lists database definition objects in the specified version number of the model.
filepath	Path and name of the file where the JSON representation of the model is written. If no file is specified, the export goes to stdout.

Examples

Export the current version (version 4) of the model. This is executed from a management instance:

```
% pwd
/sw/tten/grid/ttinstances/gridmgmt1/bin
% ttGridAdmin modelExport -current /sw/tten/grid/models/model4export.json
Model version 4 exported to /sw/tten/grid/models/model4export.json
```

Export the latest version (version 5) of the model, which is the default version to export. This is executed from a data instance:

```
% pwd
/sw/tten/grid/ttinstances/instance1/bin
% ttGridAdmin modelExport /sw/tten/grid/models/model5export.json
Model version 5 exported to /sw/tten/grid/models/model5export.json
```

Output files:

```
% pwd
/sw/tten/grid/models
% ls
model4export.json   model5export.json
```

Notes

- Metrics and logs are not exported. They exist on the active management instance but are not part of the model.

- You can execute this command from a management instance or a data instance.

- You can use modelExport to create a backup of the model.

Import a version of the model (modelImport)

The modelImport command imports a model from a JSON file (perhaps exported earlier using the modelExport command) to update the latest version of the model, or creates a script that you can use to update the model later.

```
ttGridAdmin modelImport [-script scriptpath]
                        [filepath]
```

Options

The modelImport command has the options:

Option	Description
-script *scriptpath*	Creates a script with the specified name and path. The model is not updated when you execute modelImport. Instead, you can execute the resulting script later to modify the latest version of the model to conform to the imported version. This allows you to review the changes beforehand.
	Without -script, the latest model is updated immediately.
filepath	Path and name of the JSON file from which the representation of the model is read.
	If *filepath* is not specified, input is read from stdin.

Examples

Consider a scenario where you exported the latest version (Version 5) of the model, subsequently made changes to the latest version of the model without applying them, then decided you do not want those changes after all. To undo the changes, import the file you previously exported:

```
% ttGridAdmin modelImport /sw/tten/grid/models/model5export.json
Model imported
```

Without the -script option, the model is imported immediately.

With the script option, a script is created that you can run later:

```
% ttGridAdmin modelImport /sw/tten/grid/models/model5export.json -script
/sw/tten/grid/models/modelmodscript
Script /sw/tten/grid/models/modelmodscript created.
```

Here is an example of a resulting script:

```
% pwd
/sw/tten/grid/models
% more modelmodscript
#!/bin/sh
# Created by ttGridAdmin -modelImport
TIMESTEN_HOME=/sw/tten/grid/ttinstances/gridmgmt
if [ -e $TIMESTEN_HOME/bin/ttenv.sh ]; then
 . $TIMESTEN_HOME/bin/ttenv.sh >/dev/null 2>&1
fi
# TNSNames unchanged
#Host mysys5host...
ttGridAdmin -hostCreate  mysys5host -internalAddress mysys5.example.com
-externalAddress mysys5.example.com -physicalGroup location2
ttGridAdmin installationCreate mysys5host.installslc -location
/sw/tten/grid/ttinstallations/installadc/
ttGridAdmin instanceCreate mysys5host.instance1 -installation installslc
-location /sw/tten/grid/ttinstances/ -daemonPort 20000 -csPort 21000
#Host mysys3host...
#Host mysys1host...
#Host mysys2host...
#Host mysys4host...
#Dbdef database1
#Connectable unchanged!
#Connectable unchanged!
#DbDef unchanged!
#Dbdef TTGRIDADMIN
#Connectable unchanged!
#Connectable unchanged!
```

```
#DbDef unchanged!
```

Notes

- The `modelImport` command compares the latest version of the model with the model being imported.

- The changes to the latest version of the model are not done in an atomic transaction. Each change is done in a separate transaction, so any failure will result in complications.

List model versions (modelList)

The `modelList` command lists the versions of the model, indicating when each was defined, applied, and deleted, as applicable.

```
ttGridAdmin modelList
```

Examples

```
% ttGridAdmin modelList
Version Created            Applied            Deleted
------- ------------------ ------------------ ------------------
      1 2016-10-06 12:59:26 2016-10-14 13:45:24 N/A
      2 2016-10-14 13:44:45 2016-10-14 14:33:47 N/A
      3 2016-10-14 14:33:05 2016-10-14 14:46:33 N/A
      4 2016-10-14 14:46:20 N/A                 N/A
```

Oracle Database operations

Use `ttGridAdmin` commands in this section to import or export `sqlnet.ora` configuration or TNS names entries for connecting to an Oracle database.

These are Oracle Database features that allow an application in TimesTen Scaleout to interact with an Oracle database using the `ttLoadFromOracle` utility, OCI, or Pro*C/C++.

> **Notes:**
>
> - Do not use these commands for OCI or Pro*C/C++ connections to a TimesTen Scaleout database. Entries for `tnsnames` and `sqlnet` are made automatically by TimesTen Scaleout.
>
> - The `ttLoadFromOracle` built-in procedure is for loading data from an Oracle database into TimesTen Classic or TimesTen Scaleout.
>
> - For a summary of TNS names and `sqlnet.ora`, see "Connecting to a TimesTen database from OCI" in *Oracle TimesTen In-Memory Database C Developer's Guide*.

Export a sqlnet file (SQLNetExport)

That `SQLNetExport` command exports `sqlnet.ora` configuration (that had previously been imported) from the specified version of the model, typically to a specified file.

```
ttGridAdmin SQLNetExport [-latest|-current|-version n] [filepath]
```

Options

The `SQLNetExport` command has the options:

Option	Description
`-latest`	Export `sqlnet.ora` configuration from the latest model—the model being modified and not yet applied to the grid. This is the default.
`-current`	Export `sqlnet.ora` configuration from the current model—the model most recently applied to the grid.
`-version n`	Export `sqlnet.ora` configuration from the specified version number of the model.
`filepath`	Path and name of the file that will contain the exported `sqlnet.ora` configuration. If no file is specified, the export goes to `stdout`.

Examples

This example exports `sqlnet.ora` from the latest version of the model (by default) then shows the contents of the file.

```
% ttGridAdmin SQLNetExport /sw/tten/grid/misc/sqlnet.ora
% cd /sw/tten/grid/misc
% more sqlnet.ora
# To use ezconnect syntax or tnsnames, the following entries must be
# included in the sqlnet.ora configuration.
#
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
```

Import a sqlnet file (SQLNetImport)

The SQLNetImport command imports sqlnet.ora configuration (used in communicating with an Oracle database through ttLoadFromOracle, OCI, Pro*C/C++, or ODP.NET) from the specified file into the sqlnet.ora file for the latest version of the model. This will be in place of any previously existing sqlnet.ora configuration.

```
ttGridAdmin SQLNetImport filepath
```

> **Note:** Any previous import is overwritten.

Option

The SQLNetImport command has the option:

Option	Description
filepath	Path and name of the file containing sqlnet.ora configuration to import.

Examples

```
% ttGridAdmin SQLNetImport /tmp/sqlnet.ora
SQLNet configuration file /tmp/sqlnet.ora imported
```

Notes

- This is the only way to bring sqlnet.ora configuration into the grid. Do not manually add or manipulate configuration files.

- The resulting sqlnet.ora file will be made available across all instances of the grid when you execute modelApply.

Export TNS names (TNSNamesExport)

The TNSNamesExport command exports TNS names entries (that had previously been imported) from the specified version of the model, typically to a specified file.

```
ttGridAdmin TNSNamesExport [-latest|-current|-version n] [filepath]
```

Options

The TNSNamesExport command has the options:

Option	Description
-latest	Export TNS names entries from the latest model—the model being modified and not yet applied to the grid. This is the default.
-current	Export TNS names entries from the current model—the model most recently applied to the grid.
-version *n*	Export TNS names entries from the specified version number of the model.
filepath	Path and name of the file that will contain the exported TNS names entries. If no file is specified, the export goes to stdout.

Examples

This example exports `tnsnames.ora` from the latest version of the model (by default), then shows the contents of the file.

```
% ttGridAdmin TNSNamesExport /sw/tten/grid/misc/tnsnames.ora
% cd /sw/tten/grid/misc
% more tnsnames.ora
...
  ORCL =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = www.example.com)(PORT=1630))
      (CONNECT_DATA =
        (SERVICE_NAME = orcl)
      )
    )
...
```

Import TNS names (TNSNamesImport)

The `TNSNamesImport` command imports TNS names entries (used in communicating with an Oracle database through `ttLoadFromOracle`, OCI, Pro*C/C++, or ODP.NET) from the specified file into the `tnsnames.ora` file for the latest version of the model. This will replace any previously imported TNS names entries.

```
ttGridAdmin TNSNamesImport filepath
```

Options

The `TNSNamesImport` command has the option:

Option	Description
filepath	Path and name of the file containing TNS entries to import.

Examples

```
% ttGridAdmin TNSNamesImport /tmp/tnsnames.ora
TNSNames configuration file /tmp/tnsnames.ora imported
```

Notes

- This is the only way to bring TNS names configuration into the grid. Do not manually add or manipulate configuration files.

- The resulting `tnsnames.ora` file will be made available across all instances of the grid when you execute `modelApply`.

- The `tnsnames.ora` file in the grid always contains entries for all connectables. You can add to that through `TNSNamesImport`, but you cannot remove entries other than any you have previously imported.

Physical group operations

Use the `ttGridAdmin` commands in this section to define or delete a physical group or to list physical groups in the model.

Create a physical group (physicalCreate)

The `physicalCreate` command defines a physical group in the model.

```
ttGridAdmin physicalCreate name
                          [-comment comment]
```

Options

The `physicalCreate` command has the options:

Option	Description
name	Specifies the name for the physical group in the model.
-comment *comment*	Associates a comment with the physical group object. Put the comment in quotes if there are any spaces. The comment is stored and included in output of the `physicalList` command.

Examples

```
% ttGridAdmin physicalCreate location1
PhysicalGroup location1 created.
% ttGridAdmin physicalCreate location2
PhysicalGroup location2 created.
```

Also see "Assigning hosts to physical groups" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Delete a physical group (physicalDelete)

The `physicalDelete` command removes a physical group from the model.

```
ttGridAdmin physicalDelete name
```

Options

The `physicalDelete` command has the option:

Option	Description
name	The name of the physical group to delete.

Examples

```
% ttGridAdmin physicalDelete location1
PhysicalGroup location1 deleted.
% ttGridAdmin physicalDelete location2
PhysicalGroup location2 deleted.
```

Notes

You cannot delete a physical group that is associated with any hosts.

List physical groups (physicalList)

The `physicalList` command lists all physical groups that are defined in the specified version of the model.

```
ttGridAdmin physicalList [-latest|-current|-version n]
```

Options

The `physicalList` command has the options:

Option	Description
-latest	Lists physical groups in the latest model—the model being modified and not yet applied to the grid. This is the default.
-current	Lists physical groups in the current model—the model most recently applied to the grid.
-version *n*	Lists physical groups in the specified version number of the model.

Examples

This example lists physical groups in the current version of the model, then adds another physical group and lists physical groups in the latest version (the default).

```
% ttGridAdmin physicalList -current
PhysicalGroup Comment
------------- -------
location1
location2
location3
% ttGridAdmin physicalCreate location4
PhysicalGroup location4 created.
% ttGridAdmin physicalList
PhysicalGroup Comment
------------- -------
location1
location2
location3
location4
```

Repository operations

Use `ttGridAdmin` commands in this section to create, attach, detach, or list repositories.

In TimesTen Scaleout, a *repository* is a file system directory tree used for database backups or exports or for collections of daemon logs. You specify the top-level directory when you create the repository. The contents of the directory and subdirectories of a repository, whether consisting of a backup, an export, or daemon logs, comprise a *collection*.

A repository is either mounted on each host of the grid (using NFS or equivalent), or mounted on a single host (optionally in the grid) and accessed by other hosts using `scp` (SSH copy).

For additional information, refer to "Migrating, Backing Up and Restoring Data" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Attach a repository (repositoryAttach)

The `repositoryAttach` command attaches an existing repository to the grid, making it available for use.

```
ttGridAdmin repositoryAttach name
                         -path path
                         -method mount|scp
                         [-address internalAddress]
```

Options

The `repositoryAttach` command has the options:

Option	Description
name	Name of the repository to attach.
-path *path*	Fully qualified path to the parent directory where the repository is located.
	For `-method mount`, this is the full NFS path, such as `/net/mysys2/repositories`.
	For `-method scp`, this is the full path, such as `/repositories`, on the system indicated by `-address`.
-method mount|scp	Indicates how grid instances access the repository. Supported options are `mount` or `scp`.
	▪ `mount`: The repository is accessed through an NFS mount on each grid host.
	▪ `scp`: The repository is accessed by each grid host using `scp` through passwordless SSH.
	Note: The `-method` setting for `repositoryAttach` must match the setting that was used for `repositoryCreate` when the repository was created.
-address *internalAddress*	For repositories accessed through `scp`, this option is required and indicates the DNS name or IP address of the system where the repository is located.
	Also see "Address formats" on page 4-6.

Examples

This example attaches a repository /repositories/repo1 that is located on the system mysys2, using -method scp.

```
% ttGridAdmin repositoryAttach repo1 -path /repositories -method scp
-address mysys2.example.com
Repository repo1 attached
```

Notes

This command is typically used to attach a repository that was created from another grid, so that you can restore a backup of a database from one grid into another grid. It can also be used to reattach a repository to the grid where it was created, if it was detached.

Create a repository (repositoryCreate)

The repositoryCreate command creates a new repository that will be available for the grid.

```
ttGridAdmin repositoryCreate name
                             -path path
                             -method mount|scp
                             [-address internalAddress]
```

Options

The repositoryCreate command has the options:

Option	Description
name	Specifies the name for the repository. This is the name of the directory that will be created under the parent directory specified by -path.
-path path	Fully qualified path to the parent directory where the repository is to be created. This directory must already exist on the system(s) where the repository will be located and be readable and writable by the instance administrator.
	For -method mount, this is the full NFS path, such as /net/mysys2/repositories.
	For -method scp, this is the full path, such as /repositories, on the system indicated by -address.
-method mount|scp	Indicates how grid instances access the repository. Supported options are mount or scp.
	■ mount: The repository is accessed through an NFS mount on each grid host.
	■ scp: The repository is accessed by each grid host using scp through passwordless SSH.
	Note: If you will later use repositoryAttach for the repository being created, the -method setting for repositoryAttach must match the setting you are using for repositoryCreate.
-address internalAddress	For repositories accessed through scp, this option is required and indicates the DNS name or IP address of the system where the repository is created.
	Also see "Address formats" on page 4-6.

Examples

This example creates a repository /repositories/repo1 on the system mysys2, using scp. The instance administrator must have write permission for /repositories.

```
% ttGridAdmin repositoryCreate repo1 -path /repositories -method scp
-address mysys2.example.com
Repository repo1 created
```

This example creates the repository using mount.

```
% ttGridAdmin repositoryCreate repo1 -path /repositories -method mount
Repository repo1 created
```

The repository.json file has information about the repository.

Notes

- The repository directory is created synchronously and has permissions of 700. For repositories accessed through NFS mount, the repository directory is *path*/*name*. For repositories access through scp, the repository directory is @*address*:*path*/*name*.

- The repository is available for use as soon as it is created.

- Once a repository is created with repositoryCreate, you can use repositoryAttach to access it from other grids.

Detach a repository (repositoryDetach)

The repositoryDetach command detaches (disassociates) a repository from the grid, so that it will no longer be usable from the grid.

```
ttGridAdmin repositoryDetach name
```

Options

The repositoryDetach command has the option:

Option	Description
name	Name of the repository to detach (as established when it was attached or created).

Examples

```
% ttGridAdmin repositoryDetach repo1
Repository repo1 detached
```

Notes

You can detach a repository that was created in the grid or, more typically, attached in the grid.

List repositories (repositoryList)

The repositoryList command lists repositories that are accessible (created or attached) in the grid, optionally including information about contents of the repositories—database backups, database exports, or collections of daemon logs and other information.

```
ttGridAdmin repositoryList [name]
```

```
[-contents [-details]]
```

Options

The `repositoryList` command has the options:

Option	Description
name	Name of the repository to list. If no name is specified, all repositories accessible to the grid are listed.
-contents	Show the contents of each repository listed.
-details	Show details about the contents of each repository listed (use with -contents).

Examples

This examples lists all repositories accessible to the grid (there is only one), but no contents.

```
% ttGridAdmin repositoryList
Repository Method Location            Address
---------- ------ ------------------- ------------------
repo1      scp    /repositories/repo1 mysys2.example.com
```

This example shows contents:

```
% ttGridAdmin repositoryList -contents
Repository Collection     Type   Date                    Details
---------- -------------- ------ ----------------------- ------------------
repo1      B20170222145544 Backup 2017-02-22T14:55:48.000Z Database database1
```

ttGridRollout

Description

The `ttGridRollout` utility, run from the *installation_dir*/tt18.1.4.1.0/bin directory of your TimesTen installation, creates a new grid with one database definition. The database is created and loaded, its distribution is configured, then it is opened.

The utility reads a configuration file that contains user-defined parameters and attributes for the grid you want to create. TimesTen provides a configuration template that you can copy and modify.

You can specify the shape of the grid, the hosts to use, the number of management instances (one or two), and the number of data instances, among other settings. By default, if you do not specify hosts, management instances, and data instances, then a single management instance and a number of data instances suitable for the specified shape are created on your local host.

The installation from which you run `ttGridRollout` is copied to the other hosts you specify so that additional data instances and a standby management instance can be created as desired. If you specify two management instances, the first must be on your local host.

The `ttGridRollout` utility is a wrapper for the `ttGridAdmin` utility (also using `ttInstanceCreate` for the first management instance), and actions performed by `ttGridRollout` can optionally be performed directly using `ttGridAdmin`. Once you have created a grid with `ttGridRollout`, use `ttGridAdmin` to maintain it and to make any changes.

The `ttGridRollout` utility is typically used for creating sample grids or grids that will be used during product design and evaluation.

> **Important:**
> - Execute this utility exactly as "`ttGridRollout`" (for example, not as "`ttgridrollout`").
> - The `ttGridRollout` utility does not support physical groups.

Required privilege

The user who runs this utility becomes the instance administrator of all instances created, and the user's primary user group becomes the TimesTen user group.

File system write permission is required wherever installations and instances will be created.

Usage with TimesTen Scaleout

This utility is specifically for use with TimesTen Scaleout.

Syntax

```
ttGridRollout [-h | -help | -?]
ttGridRollout [-n | -dry-run] [-wait n] [-timeout n] conf_file
```

Options

ttGridRollout has the options:

Option	Description
-h -help -?	Displays help information.
conf_file	Specifies the configuration file that contains the parameters for creating the grid and database. A read-only template, ttgrid.conf.example, is located in the *installation_dir*/tt18.1.4.1.0/grid/conf directory. You can copy and modify this file to set up your configuration.
-n | -dry-run	Displays the commands to be executed but does not execute them. Other options you specify will be reflected in the display of commands to be executed. **Note**: It is advisable to do this before executing the command.
-wait *n*	Specifies how long ttGridRollout will wait for database state changes to complete before returning. By default, there is no limit to the wait. (Database operations in TimesTen Scaleout, such as creating, loading, and opening, initiate a state change that is recorded in the active management instance of the grid. The state change is complete once the database operation has completed on each instance of the grid.)
-timeout *n*	Maximum number of seconds to wait for a long-running operation to complete. The default is 600.

> **Note:** the -wait option applies only to database operations. The -timeout option applies to any operation. These options are passed to ttGridAdmin.
>
> Also see "Command timeouts and waits" on page 4-8.

Configuration file parameters

The table that follows describes configuration parameters supported by the ttGridRollout configuration file (named ttgrid.conf by convention).

> **Important:** These parameters are required in your configuration file:
>
> - grid_name
> - dbdef_file
> - shape (optionally with data_hosts) or data_instances
> - instance_location
> - zoo_conf, unless all TimesTen instances and the membership server are on the local host

Parameter	Description
cs_connect_files	Connectable files (.connect) for client/server connectables, as desired. You can specify multiple, comma-separated .connect files. For example: `cs_connect_files = client1.connect, client2.connect` For information about connectable files, see "Create a connectable (connectableCreate)" on page 4-17.

data_hosts	List of entries for hosts to be used for data instances, in JSON format.
	
	Optionally use this with shape (and it cannot be used without shape).
	
	Do not use both data_hosts and data_instances.
	
	If you do not specify enough hosts for an *NxK* grid (see the description for shape), ttGridRollout loops back to the start of the specified host list and will place additional instances on as many hosts as necessary. If you specify too many hosts, only the first *NxK* hosts are used.
	
	This parameter supports the attributes address, externalAddress, internalAddress, installation_location, and instance_location. See "Configuration file parameter attributes" on page 4-120, including information about default values.
	
	Specifying address(es) is required—either address OR externalAddress and internalAddress.
	
	Example:
	
	`data_hosts = [`
	`{ "internalAddress":"tthost1-priv",`
	`"externalAddress":"tthost1.example.com",`
	`"installation_location":"/u01/tthost1/TimesTen" },`
	`{ "internalAddress":"tthost2-priv",`
	`"externalAddress":"tthost2.example.com",`
	`"installation_location":"/u01/tthost2/TimesTen" },`
	`{ "internalAddress":"tthost3-priv",`
	`"externalAddress":"tthost3.example.com",`
	`"installation_location":"/u01/tthost3/TimesTen" },`
	`{ "internalAddress":"tthost4-priv",`
	`"externalAddress":"tthost4.example.com",`
	`"installation_location":"/u01/tthost4/TimesTen" }`
	`]`
	
	Notes: See notes for shape.

Parameter	Description
data_instances	List of entries for data instances, in JSON format. This parameter allows you to specify data space groups, host and instance names, and daemon and client/server port numbers.
	You cannot use data_instances together with shape or data_hosts. (Specify shape, with or without data_hosts, or specify data_instances.)
	This parameter supports the attributes address, externalAddress, internalAddress, host, instance, dataSpaceGroup, daemonport, csport, installation_location, and instance_location. See "Configuration file parameter attributes" on page 4-120, including information about default values.
	The shape of the grid is determined by your dataSpaceGroup settings. If you do not specify data space groups, the grid will be Nx1, with one data space group.
	Specifying address(es) is required—either address OR externalAddress and internalAddress.
	Example:

	`data_instances = [`
	`{ "internalAddress":"tthost1-priv",`
	`"externalAddress":"tthost1.example.com",`
	`"dataspacegroup":1, "daemonport":50001, "csport":50002 },`
	`{ "internalAddress":"tthost2-priv",`
	`"externalAddress":"tthost2.example.com",`
	`"dataspacegroup":1, "daemonport":50001, "csport":50002 },`
	`{ "internalAddress":"tthost3-priv",`
	`"externalAddress":"tthost3.example.com",`
	`"dataspacegroup":2, "daemonport":50001, "csport":50002 },`
	`{ "internalAddress":"tthost4-priv",`
	`"externalAddress":"tthost4.example.com",`
	`"dataspacegroup":2, "daemonport":50001, "csport":50002 }`
	`]`

	Notes:
	■ A host object is created for each instance, except where two data instances are specified to be on the same system and in the same dataspace group, in which case they will share the same host object.
	■ An installation object is created for each host object. Multiple installation objects can point to the same physical installation.
dbdef_file	Database definition file (.dbdef). This is required.
	Directories are created on each host as necessary for the DataStore and LogDir locations.
	For information about database definition files, see "Create a database definition (dbdefCreate)" on page 4-24.
direct_connect_files	Connectable files (.connect) for additional direct connectables, as desired, beyond the connectable that is automatically created when the database is created. You can specify multiple, comma-separated .connect files. For example:
	`direct_connect_files = mydbcfg1.connect, mydbcfg2.connect`
	For information about connectable files, see "Create a connectable (connectableCreate)" on page 4-17.
grid_name	The desired name of the grid. This is required.

Parameter	Description
init_script	A SQL script (ttIsql script) for ttGridRollout to execute on the database after rolling out the grid (using the first data instance that was created). For example, the script may include SQL statements to create database users and schemas.
installation_location	Path to the parent directory where you want to put the TimesTen installation on systems where the standby management instance (if applicable) and data instances are located. The tt18.1.4.1.0 directory is directly under this location. The directory is created on each host as necessary. If you specify an existing location, the directory must be empty.
	The default is to use the same location as for the installation on the local host, from which ttGridRollout is run.
	This location is used throughout the grid, except where you override it for a particular host or instance by setting the installation_location attribute of the data_hosts, data_instances, or mgmt_instances parameter.
instance_config	A file for custom configuration of data instances, consisting of *name=value* pairs for any settings you want to add to the instance configurations.
	This is accomplished using the ttGridAdmin instanceConfigImport command. Also see "Import instance configuration attributes (instanceConfigImport)" on page 4-82.
instance_location	Path to the parent directory for TimesTen instances (data and management). This is required. For each instance, the *timesten_home* directory will be named *instancename* under this location. The directory is created on each host as necessary.
	This location is used throughout the grid, except where you override it for a particular host or instance by setting the instance_location attribute of the data_hosts, data_instances, or mgmt_instances parameter.
mgmt_instances	List of entries for management instances, in JSON format. The first entry must be on the local host and will be the active management instance. The second entry (if applicable) must be on a different system and will be the standby management instance.
	If you do not set mgmt_instances, ttGridRollout creates one management instance on the local host.
	This parameter supports the attributes address, externalAddress, internalAddress, host, instance, daemonport, csport, mgmtport, installation_location, and instance_location. See "Configuration file parameter attributes" on page 4-120, including information about default values.
	Specifying address(es) is required—either address OR externalAddress and internalAddress.
	Example:
	``` mgmt_instances = [{ "internalAddress":"tthost1-priv", "externalAddress":"tthost1.example.com" }, { "internalAddress":"tthost2-priv", "externalAddress":"tthost2.example.com" }] ```
	Notes:
	■ A host object and an installation object are created for each instance.

Parameter	Description
shape	The desired shape of the grid, *NxK*, where:
	■ *N* is the number of instances in each data space group.
	■ *K* is the K-factor (replication factor) of the grid, which is by definition the number of data space groups (1 or 2).
	Either specify shape, with or without data_hosts, or specify data_instances. If you use shape without data_hosts, all TimesTen instances are placed on the local host.
	When you specify shape for an *NxK* grid, *NxK* instances will be created (such eight instances for a 4x2 grid). The first *N* instances will be in data group 1, and for k=2 the next *N* data instances will be in data space group 2.
	Notes:
	■ A host object is created for each instance, except where two data instances will be on the same system and in the same dataspace group, in which case they will share the same host object.
	■ If you specify host addresses as DNS names, default host object names are according to the addresses (such as mysys1 for an address mysys1.example.com, with _2 appended if there is a second host object on the same system, or mysys1_mgmt for the host of a management instance). If you specify addresses as IP addresses, default host object names are host_*n* sequentially.
	■ An installation object is created for each host object. Multiple installation objects can point to the same physical installation.
	■ As ttGridRollout creates instances, it names them instance1, instance2, instance3, and so on.
sqlnet_config	SQL*Net configuration file (used in communicating with an Oracle database through ttLoadFromOracle, OCI, Pro*C/C++, or ODP.NET).
	Through the ttGridAdmin SQLNetImport command, ttGridRollout applies the specified SQL*Net configuration on all data instances. Also see "Export a sqlnet file (SQLNetExport)" on page 4-105.
tnsnames_config	TNS names configuration file (used in communicating with an Oracle database through ttLoadFromOracle, OCI, Pro*C/C++, or ODP.NET).
	Through the ttGridAdmin TNSNamesImport command, ttGridRollout applies the specified TNS names configuration on all data instances. Also see "Import TNS names (TNSNamesImport)" on page 4-107.
zoo_conf	Apache ZooKeeper membership service client configuration file. This parameter is required unless all management instances, data instances, and the ZooKeeper membership server will be on your local host.
	For examples of ZooKeeper client configuration files, see "Membership operations" on page 4-97. For details on how to configure ZooKeeper as a membership service, see "Using Apache ZooKeeper as the membership service" in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*.
	If you do not specify this parameter, ttGridRollout assumes that a ZooKeeper server already runs on the local host using the default client port setting, 2181.

Configuration file parameter attributes

The ttGridRollout configuration parameters support these attributes. Refer to the preceding table of parameters to see which attributes are supported by each parameter.

Attribute	Description
address	DNS name or IP address of the system for both external and internal communications, if a single address is used. Either use address or use internalAddress and externalAddress. Setting -address *xxx* is exactly equivalent to setting -internalAddress *xxx* and -externalAddress *xxx*.
	This option takes one name or address only, and a specified name must resolve to one IP address or to multiple IP addresses on the same network segment.
	If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid.
	Note: Using a single address is not recommended for production environments.
	Also see "Address formats" on page 4-6.
csport	Port for client/server connections.
	If this is not specified for a data instance, ttGridRollout uses an available port between 46337 and 46997.
	If this is not specified for a management instance, ttGridRollout attempts to use the TimesTen default client/server port, 6625.
daemonport	Port for TimesTen daemon communications.
	If this is not specified for a data instance, ttGridRollout uses an available port between 46337 and 46997.
	If this is not specified for a management instance, ttGridRollout attempts to use the TimesTen default daemon port, 6624.
dataSpaceGroup	Desired data space group (1 or 2). The default is data space group 1.
	If you use the data_instances parameter, you can use this attribute to specify the data space group for the instance.
externalAddress	DNS name or IP address of the system for external communications (outside the grid) for client/server connections. Either use address or use internalAddress and externalAddress. Setting -internalAddress *xxx* and -externalAddress *xxx* is exactly equivalent to setting -address *xxx*.
	This option takes one name or address only, but a name may resolve to one or more IP addresses.
	If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid.
	Also see "Address formats" on page 4-6.
host	Desired name of the host object in the grid model.
	Note: If you specify host addresses as DNS names, default host object names are according to the addresses (such as mysys1 for an address mysys1.example.com, with _2 appended if there is a second host object on the same system, or mysys1_mgmt for the host of a management instance). If you specify addresses as IP addresses, default host object names are host_*n* sequentially.

Attribute	Description
installation_location	Overrides the grid-wide installation_location setting for a host or instance. See installation_location under "Configuration file parameters" on page 4-115 for additional information.
instance	If you use the data_instances or mgmt_instances parameter (as appropriate), you can use this attribute to specify the instance name. Alternatively, for data instances, ttGridRollout names the instances instance1, instance2, instance3, and so on. Alternatively, for management instances, ttGridRollout names the instances *gridname*_mgmt and *gridname*_mgmt2 (if there is a second management instance).
instance_location	Overrides the grid-wide instance_location setting for a host or instance. See instance_location under "Configuration file parameters" on page 4-115 for additional information.
internalAddress	DNS name or IP address for internal communications (within the grid). Either use address or use internalAddress and externalAddress. Setting -internalAddress *xxx* and -externalAddress *xxx* is exactly equivalent to setting -address *xxx*. This option takes one name or address only, and a specified name must resolve to one IP address or to multiple IP addresses on the same network segment. If host names from /etc/hosts are being used, the /etc/hosts files on all instances in the grid must contain identical entries for all hosts in the grid. Also see "Address formats" on page 4-6.
mgmtport	Port for management instance communications. If this is not specified, ttGridRollout attempts to use the TimesTen default management port, 3754.

Examples

This sections provides three ttGridRollout examples with various types of configuration:

- shape parameter without data_hosts to configure a 2x2 grid with one management instance, all on the local host

- shape parameter with data_hosts to configure a 2x2 grid with one management instance on four systems, with the management instance and a data instance on the local host

- data_instances parameter to configure a 3x2 grid with two management instances on eight systems, with the first management instance on the local host

Each example includes the configuration, the dry run output showing the ttGridAdmin commands to be executed, and portions of the execution output. In each example, mysys1 is the local host. Dry run output is edited for readability. (The ttInstanceCreate utility, to create the first management instance, is executed through the full path to the installation bin directory; ttGridAdmin is executed through the full path to the TimesTen ttenv environment setup script in the first management instance bin directory.)

Additional examples are in "Deploy a grid and database" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

shape without data_hosts: This scenario is convenient for standalone development.

Configuration:

```
dbdef_file = /sw/tten/dbdef/database1.dbdef
shape = 2x2
zoo_conf = /sw/tten/zkconfig/membership.conf
grid_name = grid1
instance_location = /sw/tten/grid1/ttinstances
```

Dry run:

```
% ./ttGridRollout -dry-run /sw/tten/gridconfig/ttgrid1.conf
ttInstanceCreate -grid -location /sw/tten/grid1/ttinstances -name grid1_mgmt
ttGridAdmin gridCreate grid1 -k 2 -host mysys1_mgmt -address mysys1
 -membershipConfig /sw/tten/zkconfig/membership.conf
ttGridAdmin hostCreate mysys1 -address mysys1 -dataspacegroup 1
ttGridAdmin installationCreate mysys1 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys1_2 -address mysys1 -dataspacegroup 2
ttGridAdmin installationCreate mysys1_2 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin instanceCreate mysys1.instance1 -location /sw/tten/grid1/ttinstances
 -daemonport 46337 -csport 46338
ttGridAdmin instanceCreate mysys1.instance2 -location /sw/tten/grid1/ttinstances
 -daemonport 46339 -csport 46340
ttGridAdmin instanceCreate mysys1_2.instance3 -location /sw/tten/grid1/ttinstances
 -daemonport 46341 -csport 46342
ttGridAdmin instanceCreate mysys1_2.instance4 -location /sw/tten/grid1/ttinstances
 -daemonport 46343 -csport 46344
ttGridAdmin dbdefCreate /sw/tten/dbdef/database1.dbdef
ttGridAdmin modelApply
ttGridAdmin dbCreate -wait database1
ttGridAdmin dbDistribute database1 -add all -apply
ttGridAdmin dbOpen -wait database1
```

Execution:

```
% ./ttGridRollout /sw/tten/gridconfig/ttgrid1.conf
INFO: Generating data_instances for 2x2 Grid
data_instances = [
 { "address":"mysys1", "dataspacegroup":1 },
 { "address":"mysys1", "dataspacegroup":1 },
 { "address":"mysys1", "dataspacegroup":2 },
 { "address":"mysys1", "dataspacegroup":2 }
]
INFO: Checking Zookeeper on zk1!2181 -- OK
INFO: Checking Zookeeper on zk2!2181 -- OK
INFO: Checking Zookeeper on zk3!2181 -- OK
INFO: Checking the address for the management database -- OK
INFO: Checking connectivity to mysys1 -- OK


==============================================================================

...

==============================================================================
4-instance (2x2) grid successfully created.

Management Instance Location
---------------------------
- mysys1:/sw/tten/grid1/ttinstances/grid1_mgmt
```

```
...

Data Instance Locations
-----------------------
- mysys1.instance1 ==> mysys1:/sw/tten/grid1/ttinstances/instance1
- mysys1.instance2 ==> mysys1:/sw/tten/grid1/ttinstances/instance2
- mysys1_2.instance3 ==> mysys1:/sw/tten/grid1/ttinstances/instance3
- mysys1_2.instance4 ==> mysys1:/sw/tten/grid1/ttinstances/instance4

...
```

shape with data_hosts: This scenario is useful for initial testing on multiple systems.

Configuration:

```
dbdef_file = /sw/tten/dbdef/database1.dbdef
shape = 2x2
zoo_conf = /sw/tten/zkconfig/membership.conf
grid_name = grid1
instance_location = /sw/tten/grid1/ttinstances
data_hosts = [
  { "internalAddress":"mysys1-i", "externalAddress":"mysys1.example.com" },
  { "internalAddress":"mysys2-i", "externalAddress":"mysys2.example.com" },
  { "internalAddress":"mysys3-i", "externalAddress":"mysys3.example.com" },
  { "internalAddress":"mysys4-i", "externalAddress":"mysys4.example.com" }
]
```

Dry run:

```
% ./ttGridRollout -dry-run /sw/tten/gridconfig/ttgrid1.conf
ttInstanceCreate -grid -location /sw/tten/grid1/ttinstances -name grid1_mgmt
ttGridAdmin gridCreate grid1 -k 2 -host mysys1_mgmt -address mysys1
 -membershipConfig /sw/tten/zkconfig/membership.conf
ttGridAdmin hostCreate mysys1 -externaladdress mysys1.example.com
 -internaladdress mysys1-i -dataspacegroup 1
ttGridAdmin installationCreate mysys1 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys2 -externaladdress mysys2.example.com
 -internaladdress mysys2-i -dataspacegroup 1
ttGridAdmin installationCreate mysys2 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys3 -externaladdress mysys3.example.com
 -internaladdress mysys3-i -dataspacegroup 2
ttGridAdmin installationCreate mysys3 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys4 -externaladdress mysys4.example.com
 -internaladdress mysys4-i -dataspacegroup 2
ttGridAdmin installationCreate mysys4 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin instanceCreate mysys1.instance1 -location /sw/tten/grid1/ttinstances
 -daemonport 46337 -csport 46338
ttGridAdmin instanceCreate mysys2.instance2 -location /sw/tten/grid1/ttinstances
 -daemonport 46339 -csport 46340
ttGridAdmin instanceCreate mysys3.instance3 -location /sw/tten/grid1/ttinstances
 -daemonport 46341 -csport 46342
ttGridAdmin instanceCreate mysys4.instance4 -location /sw/tten/grid1/ttinstances
 -daemonport 46343 -csport 46344
ttGridAdmin dbdefCreate /sw/tten/dbdef/database1.dbdef
ttGridAdmin modelApply
```

```
ttGridAdmin dbCreate -wait database1
ttGridAdmin dbDistribute database1 -add all -apply
ttGridAdmin dbOpen -wait database1
```

Execution:

```
% ./ttGridRollout /sw/tten/gridconfig/ttgrid1.conf
INFO: Generating data_instances for 2x2 Grid
data_instances = [
 { "externaladdress":"mysys1.example.com", "internaladdress":"mysys1-i",
 "dataspacegroup":1 },
 { "externaladdress":"mysys2.example.com", "internaladdress":"mysys2-i",
 "dataspacegroup":1 },
 { "externaladdress":"mysys3.example.com", "internaladdress":"mysys3-i",
 "dataspacegroup":2 },
 { "externaladdress":"mysys4.example.com", "internaladdress":"mysys4-i",
 "dataspacegroup":2 }

]
INFO: Checking Zookeeper on zk1!2181 -- OK
INFO: Checking Zookeeper on zk2!2181 -- OK
INFO: Checking Zookeeper on zk3!2181 -- OK
INFO: Checking the address for the management database -- OK
INFO: Checking connectivity to mysys1 -- OK
INFO: Checking connectivity to mysys1-i -- OK
INFO: Checking connectivity to mysys2-i -- OK
INFO: Checking connectivity to mysys3-i -- OK
INFO: Checking connectivity to mysys4-i -- OK


===============================================================================

...


===============================================================================
4-instance (2x2) grid successfully created.

Management Instance Location
----------------------------
- mysys1:/sw/tten/grid1/ttinstances/grid1_mgmt


...


Data Instance Locations
-----------------------
- mysys1.instance1 ==> mysys1-i:/sw/tten/grid1/ttinstances/instance1
- mysys2.instance2 ==> mysys2-i:/sw/tten/grid1/ttinstances/instance2
- mysys3.instance3 ==> mysys3-i:/sw/tten/grid1/ttinstances/instance3
- mysys4.instance4 ==> mysys4-i:/sw/tten/grid1/ttinstances/instance4


...
```

data_instances: This scenario is useful for more realistic proof-of-concept testing.

Configuration:

```
dbdef_file = /sw/tten/dbdef/database1.dbdef
zoo_conf = /sw/tten/zkconfig/membership.conf
grid_name = grid1
instance_location = /sw/tten/grid1/ttinstances
data_instances = [
```

```
    { "internalAddress":"mysys3-i", "externalAddress":"mysys3.example.com",
  "dataspacegroup":1, "daemonport":50001, "csport":50002 },
    { "internalAddress":"mysys4-i", "externalAddress":"mysys4.example.com",
  "dataspacegroup":1, "daemonport":50001, "csport":50002 },
    { "internalAddress":"mysys5-i", "externalAddress":"mysys5.example.com",
  "dataspacegroup":1, "daemonport":50001, "csport":50002 },
    { "internalAddress":"mysys6-i", "externalAddress":"mysys6.example.com",
  "dataspacegroup":2, "daemonport":50001, "csport":50002 },
    { "internalAddress":"mysys7-i", "externalAddress":"mysys7.example.com",
  "dataspacegroup":2, "daemonport":50001, "csport":50002 },
    { "internalAddress":"mysys8-i", "externalAddress":"mysys8.example.com",
  "dataspacegroup":2, "daemonport":50001, "csport":50002 }
]
mgmt_instances = [
    { "internalAddress":"mysys1-i", "externalAddress":"mysys1.example.com" },
    { "internalAddress":"mysys2-i", "externalAddress":"mysys2.example.com" }
]
```

Dry run:

```
% ./ttGridRollout -dry-run /sw/tten/gridconfig/ttgrid1.conf
ttInstanceCreate -grid -location /sw/tten/grid1/ttinstances -name grid1_mgmt
ttGridAdmin gridCreate grid1 -k 2 -host mysys1-i_mgmt -internalAddress mysys1-i
 -externalAddress mysys1.example.com -membershipConfig
 /sw/tten/zkconfig/membership.conf
ttGridAdmin hostCreate mysys2-i_mgmt -internalAddress mysys2-i -externalAddress
 mysys2.example.com
ttGridAdmin installationCreate mysys2-i_mgmt -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin instanceCreate mysys2-i_mgmt.grid1_mgmt2 -location
 /sw/tten/grid1/ttinstances -type management
ttGridAdmin hostCreate mysys3 -externaladdress mysys3.example.com
 -internaladdress mysys3-i -dataspacegroup 1
ttGridAdmin installationCreate mysys3 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys4 -externaladdress mysys4.example.com
 -internaladdress mysys4-i -dataspacegroup 1
ttGridAdmin installationCreate mysys4 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys5 -externaladdress mysys5.example.com
 -internaladdress mysys5-i -dataspacegroup 1
ttGridAdmin installationCreate mysys5 -location
/sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys6 -externaladdress mysys6.example.com -internaladdress
 mysys6-i -dataspacegroup 2
ttGridAdmin installationCreate mysys6 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys7 -externaladdress mysys7.example.com
 -internaladdress mysys7-i -dataspacegroup 2
ttGridAdmin installationCreate mysys7 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin hostCreate mysys8 -externaladdress mysys8.example.com
 -internaladdress mysys8-i -dataspacegroup 2
ttGridAdmin installationCreate mysys8 -location
 /sw/tten/grid1/ttinstall/installation1
ttGridAdmin instanceCreate mysys3.instance1 -location /sw/tten/grid1/ttinstances
 -daemonport 50001 -csport 50002
ttGridAdmin instanceCreate mysys4.instance2 -location /sw/tten/grid1/ttinstances
 -daemonport 50001 -csport 50002
ttGridAdmin instanceCreate mysys5.instance3 -location /sw/tten/grid1/ttinstances
 -daemonport 50001 -csport 50002
```

```
ttGridAdmin instanceCreate mysys6.instance4 -location /sw/tten/grid1/ttinstances
 -daemonport 50001 -csport 50002
ttGridAdmin instanceCreate mysys7.instance5 -location /sw/tten/grid1/ttinstances
 -daemonport 50001 -csport 50002
ttGridAdmin instanceCreate mysys8.instance6 -location /sw/tten/grid1/ttinstances
 -daemonport 50001 -csport 50002
ttGridAdmin dbdefCreate /sw/tten/dbdef/database1.dbdef
ttGridAdmin modelApply
ttGridAdmin dbCreate -wait database1
ttGridAdmin dbDistribute database1 -add all -apply
ttGridAdmin dbOpen -wait database1
```

Execution:

```
% ./ttGridRollout /sw/tten/gridconfig/ttgrid1.conf
INFO: Checking Zookeeper on zk1!2181 -- OK
INFO: Checking Zookeeper on zk2!2181 -- OK
INFO: Checking Zookeeper on zk3!2181 -- OK
INFO: Checking the address for the management database -- OK
INFO: Checking connectivity to mysys1-i -- OK
INFO: Checking connectivity to mysys2-i -- OK
INFO: Checking connectivity to mysys3-i -- OK
INFO: Checking connectivity to mysys4-i -- OK
INFO: Checking connectivity to mysys5-i -- OK
INFO: Checking connectivity to mysys6-i -- OK
INFO: Checking connectivity to mysys7-i -- OK
INFO: Checking connectivity to mysys8-i -- OK


===============================================================================

...

===============================================================================
6-instance (3x2) grid successfully created.

Management Instance Locations
-----------------------------
- mysys1-i:/sw/tten/grid1/ttinstances/grid1_mgmt
- mysys2-i:/sw/tten/grid1/ttinstances/grid1_mgmt2

...

Data Instance Locations
-----------------------
- mysys3.instance1 ==> mysys3-i:/sw/tten/grid1/ttinstances/instance1
- mysys4.instance2 ==> mysys4-i:/sw/tten/grid1/ttinstances/instance2
- mysys5.instance3 ==> mysys5-i:/sw/tten/grid1/ttinstances/instance3
- mysys6.instance4 ==> mysys6-i:/sw/tten/grid1/ttinstances/instance4
- mysys7.instance5 ==> mysys7-i:/sw/tten/grid1/ttinstances/instance5
- mysys8.instance6 ==> mysys8-i:/sw/tten/grid1/ttinstances/instance6

...
```

5

Utilities

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

- Overview
- Utilities List

For information about utilities that are only supported in TimesTen Scaleout see Chapter 4, "TimesTen Scaleout 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.

Utilities List

Utilities listed in Table 5–1 are described in Chapter 4, "TimesTen Scaleout Utilities".

Utilities listed in Table 5–2 are described in this chapter.

Table 5–1   Utilities supported only in TimesTen Scaleout descriptions

Name	Description
ttGridAdmin	Administers a TimesTen Scaleout grid.
ttGridRollout	Creates a new grid and database.

Table 5–2   Other utilities descriptions

Name	Description	Usage with TimesTen Scaleout
ttAdmin	Specifies or changes database policies.	No

Table 5–2   (Cont.) Other utilities descriptions

Name	Description	Usage with TimesTen Scaleout
ttAdoptStores	Moves databases from a TimesTen installation to a new TimesTen installation of the same major release, but a different minor release.	No
ttBackup	Creates a backup copy of a database that can be restored at a later time using the ttRestore utility.	No
ttBulkCp	Copies data between TimesTen tables and ASCII files.	No
ttCapture	Captures information about the state of TimesTen.	No
ttCheck	Performs internal consistency checking within a TimesTen database.	No
ttCWAdmin	Manages TimesTen active standby pairs that take advantage of the high availability framework of Oracle Clusterware.	No
ttDaemonAdmin	Starts and stops the TimesTen main daemon and Server.	No
ttDaemonLog	Controls and displays daemon log messages.	No
ttDestroy	Destroys a database including all checkpoint files, transaction logs and daemon catalog entries.	No
ttInstallationCheck	Examines all files in an installation of TimesTen and generates a signature for the installation,	Yes
ttInstallDSN	Generates a Windows client DSN for one or more entries listed in the provided input file and installs them into the ODBC Panel as a System DSN.	Yes
ttInstanceCreate	Create a new TimesTen instance.	Yes
ttInstanceDestroy	Destroys an existing TimesTen instance.	Yes
ttInstanceModify	Modifies certain attributes of an instance.	Yes
ttIsql	Executes SQL statements interactively.	Yes

Table 5–2   (Cont.) Other utilities descriptions

Name	Description	Usage with TimesTen Scaleout
ttMigrate	Saves and restores TimesTen objects.	Yes, only for migrating from TimesTen Classic to TimesTen Scaleout.
ttRepAdmin	Displays, sets, modifies and monitors existing replication definitions and status.	No
ttRestore	Creates a database from a backup that has been created using the ttBackup utility.	No
ttSchema	Prints out the schema, or selected objects, of a database.	Yes
ttSize	Estimates the amount of space that a given table, including any views in the database will consume when the table grows to include a specified number of rows.	Yes
ttStats	Monitors database metrics or takes and compares snapshots of metrics.	Yes
ttStatus	Displays information that describes the current state of TimesTen.	No
ttTail	Fetches TimesTen internal trace information from a database and displays it to stdout.	No
ttTraceMon	Enables and disables the TimesTen internal tracing facilities.	No
ttUser	Prompts for a password and returns an encrypted password.	None
ttVersion	lists the TimesTen release information.	Yes
ttXactAdmin	Lists ownership, status, log and lock information for each outstanding transaction.	No
ttXactLog	Displays a formatted dump of the contents of a TimesTen transaction log.	No

ttAdmin

Description

This utility enables you to:

- Get information about `ttAdmin` options, version, and settings. See "Help, version, and query options" on page 5-6.

- Specify settings for database loading and RAM loading policies. See "Set database loading policies" on page 5-8.

- Open or close a database to user connections. See "Open or close a database" on page 5-10.

- Execute a forced disconnect operation for existing database connections. See "Force disconnect" on page 5-11.

- 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. See "Set cache policies" on page 5-13.

- Specify settings to automatically or manually start and stop replication agents for specified databases. See "Set replication policies" on page 5-16.

Required privilege

This utility requires no privileges to query the database.

Open and close options require the instance administrator privilege.

Replication options require the ADMIN privilege.

Cache options require the CACHE_MANAGER privilege.

All other options require the ADMIN privilege.

Usage with TimesTen Scaleout

This utility is not supported in TimesTen Scaleout.

Syntax

```
ttAdmin {-h | -help | -?}
ttAdmin {-V | -version}
ttAdmin -query {-connStr connection_string | DSN}

ttAdmin [-ramPolicy always | manual | inUse [-ramGrace secs]]
[-ramLoad | -ramUnload]
[-autoreload | -noautoreload]
{-connStr connection_string | DSN}

ttAdmin [-open | -close]
[-ramLoad]
{-connStr connection_string | DSN}

ttAdmin [-disconnect urgency [granularity]] {-connStr connection_string | DSN}
 urgency:     -transactional | -immediate | -abort
 granularity: -users | -unload
```

```
ttAdmin [-repPolicy always | manual | norestart]
[-repStart | -repStop]
[-repQueryThresholdSet secs]
[-repQueryThresholdGet]
{-connStr connection_string | DSN}

ttAdmin [
[-cacheUidGet] |
[-cacheUidPwdSet -cacheUid uid [-cachePwd pwd]] |
[-cachePolicy always | manual | norestart] |
[-cacheStart] | [-cacheStop [-stopTimeout secs]]
]
{-connStr connection_string | DSN}
```

Notes

These notes apply to all modes of `ttAdmin` usage.

Always specify the TimesTen database location as a full path. If a relative path is specified, TimesTen would look relative to the working directory of the daemon, *timesten_home*/info.

For details on environment variables that you may want to set, see "Environment variables" in *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide*.

This utility is supported only for TimesTen Data Manager DSNs. It is not supported for TimesTen Client DSNs.

See also

ttStatus
ttCachePolicySet
ttCachePolicyGet
ttCacheUidPwdSet
ttCacheStart
ttCacheStop

Help, version, and query options

Options

ttAdmin has these options for help, version, and settings information:

Option	Description
-h -help -?	Prints usage information, syntax, and option descriptions.
-V | -version	Prints the TimesTen release number of ttAdmin and exits.
-query	Displays a summary of the policy settings for the specified database.
-connStr *connection_string*	For -query, an ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
DSN	For -query, an ODBC data source name of the database to be administered.

Examples

To get the version of ttAdmin:

```
% ttAdmin -version
TimesTen Release 18.1.4.1.0
```

To get help for ttAdmin:

```
% ttAdmin -help
Usage:
  ttAdmin [-h | -help | -?]
  ttAdmin [-V | -version]
  ttAdmin [-ramUnload | -ramLoad [-open | -close]]
          [-ramPolicy always | manual | inUse [-ramGrace <secs>]]
          [-autoreload | -noautoreload]
          [-disconnect { -transactional | -immediate | -abort }
            [-users | -unload]
          ]
          [-repPolicy always | manual | norestart]
          [-repStart | -repStop]
          [-repQueryThresholdSet <seconds>]
          [-repQueryThresholdGet]
          [
            [-cacheUidGet] |
            [-cacheUidPwdSet -cacheUid <uid> [-cachePwd <pwd>]] |
            [-cachePolicy always | manual | norestart] |
            [-cacheStart] |
            [-cacheStop [-stopTimeout <secs>]]
          ]
          [-query]
          {<DSN> | [-connstr] <connStr>}

options:
[...Option descriptions not shown...]
```

To display a summary of the policy settings for the database1 DSN:

```
% ttAdmin -query database1
```

```
RAM Residence Policy          : inUse
Replication Agent Policy      : manual
Replication Manually Started  : False
Cache Agent Policy            : manual
Cache Agent Manually Started  : False
Database State                : Open
```

Set database loading policies

Options

`ttAdmin` has these options for database loading and RAM loading:

Option	Description
`-connStr connection_string`	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
`DSN`	An ODBC data source name of the database to be administered.
`-autoreload │ -noautoreload`	If set to `-autoreload` (default), TimesTen reloads the database after an invalidation.
	if set to `-noautoreload`, TimesTen does not automatically reload the database after an invalidation.
`-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 `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.
	`manual` - Specifies that the database is only to be loaded in system RAM when explicitly loaded by the user (using the `ramLoad` option). This is the recommended RAM policy, because it avoids unnecessary database loading or unloading.
	`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 policy is not supported in TimesTen Scaleout.
	`always` - Specifies that the database should remain in system RAM all the time.
`-ramUnload`	Valid only when `-ramPolicy` is set to `manual`. Causes the database to be unloaded from RAM.

Examples

To manually control whether database1 is loaded into RAM and to load it now, use the following.

```
% ttAdmin -ramPolicy manual -ramLoad database1
```

The database1 database is typically always resident in RAM. However, it is not being used at a given time and should be loaded only when applications are connected to it. To change the RAM policy:

```
ttAdmin -ramPolicy inUse database1
```

Now assume database1 is not always in use. Permanently loading it into RAM would unnecessarily use 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:

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

Some performance-sensitive applications use a database referred to by DSN database1. 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 accomplish this:

```
% ttAdmin -ramPolicy always database1
```

Notes

Setting RAM policy to *inUse* for production systems with large databases may cause performance issues having the database unload and reload unexpectedly.

The only -ramPolicy value supported for temporary databases is -ramPolicy manual with the -ramLoad option specified at the same time.

If ttAdmin is used with the -ramPolicy always option, a persistent system connection is created on the database.

RAM policy always conflicts with forced disconnect granularity unload, described in "Force disconnect" on page 5-11. Using both simultaneously results in an error and the disconnect request being ignored.

The *always* RAM policy should be used with caution. When failures occur, it may not be beneficial to have your database automatically reload. In addition, it may affect system startup performance if all databases load at the same time when your system boots.

See "Specifying a RAM policy" in the *Oracle TimesTen In-Memory Database Operations Guide* for more details.

Open or close a database

Options

ttAdmin has these options for opening or closing a database:

Option	Description
-close	Closes a database to user connections.
	When a database is closed to user connections, new connection attempts will fail, but existing connections are unaffected.
-connStr *connection_string*	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
DSN	An ODBC data source name of the database to be administered.
-open	Opens a database to user connections.
	A database is open to user connections by default upon creation.
-ramLoad	Valid only when -ramPolicy is manual. Causes the database to be loaded into RAM.

Examples

To open the database1 DSN:

```
% ttAdmin -open database1
```

To load and open the database1 DSN from an unloaded and closed state:

```
% ttAdmin -ramLoad -open database1
```

To close the database1 DSN:

```
% ttAdmin -close database1
```

To load and close the database1 DSN from an unloaded and open state:

```
% ttAdmin -ramLoad -close database1
```

Notes

A database remains closed or open to user connections regardless of its loaded state or RAM policy, unless its closed or open state is modified through ttAdmin or other utilities like ttRestore or ttRepAdmin.

If the -open or -close options are used in conjunction with any option other than -ramLoad, ttAdmin returns an error.

Trying to close a closed database or open an open database returns an error.

Force disconnect

Options

ttAdmin has these options for forced disconnect:

Option	Description
-connStr *connection_string*	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
DSN	An ODBC data source name of the database to be administered.
-disconnect *urgency* [*granularity*]	Asynchronously disconnects connected applications from the database, optionally including those that are idle or unresponsive.
	Acceptable values for *urgency* (required) are:
	-transactional - Allows any open transactions to be committed or rolled back before disconnecting. Does not affect idle connections.
	-immediate - Rolls back any open transactions before immediately disconnecting. This also disconnects idle connections.
	-abort - Aborts all direct mode application processes and client/server processes (ttcserver) in order to disconnect them.
	Note: A recommended best practice is to run -disconnect twice, as necessary. First run it in transactional urgency level. Then, after allowing some time, if not all connections have been closed yet, run it in immediate urgency level. Use ttStatus to confirm whether connections have been closed.
	Use abort urgency level only as a last resort if transactional and immediate levels do not result in all connections being closed. **Abort may result in loss of data**. Abort abruptly causes every user and ttcserver process connected to the database to exit. This may result in lost transactions.
	Acceptable values for *granularity* are:
	-users (default) - Disconnects every user connection to the database. This is useful for administrators who want to perform database maintenance.
	-unload - Disconnects every connection to the database, including the subdaemon. This cleanly unloads the database.
	Note: RAM policy always, described in "Set database loading policies" on page 5-8, conflicts with forced disconnect granularity unload. Using both simultaneously results in an error and the disconnect request being ignored.

Examples

This sample script uses -disconnect to disconnect all connections to database1, first using transactional urgency level then immediate urgency level.

```
#!/bin/sh
```

```
# close the databae
ttAdmin -close database1

# disconnect users and unload the database
ttAdmin -disconnect -transactional -unload database1

# wait 10 seconds for the disconnects to finish
COUNT = 0
while [ ttStatus | grep "pending disconnection" ] || [ $COUNT -ne 10 ]
do
  sleep 1
  COUNT=$((COUNT+1))
done

# increase urgency to immediate
if [ ttStatus | grep "pending disconnection" ]; then
  ttAdmin -disconnect -immediate -unload database1
fi
```

Use `ttStatus` to check progress. During forced disconnect, output indicates the pending disconnections:

```
TimesTen status report as of Wed Jul 18 09:55:20 2018

Daemon pid 10457 port 6627 instance user1
TimesTen server pid 10464 started on port 6629
------------------------------------------------------------------------
------------------------------------------------------------------------
Closed to user connections
Data store /databases/database1
Daemon pid 10457 port 6627 instance user1
TimesTen server pid 10464 started on port 6629
There are 14 connections to the data store, ***14 pending disconnection***
Shared Memory KEY 0x0210679b ID 949092358
PL/SQL Memory KEY 0x0310679b ID 949125127 Address 0x5000000000
Type          PID     Context           Connection Name          ConnID
Process       10484   0x00007f3ddfeb4010  database1 1
...
```

Notes

To enable the capability for forced disconnect, use the TimesTen connection attribute setting `ForceDisconnectEnabled=1`. See "ForceDisconnectEnabled" on page 2-37.

The `-disconnect` option is asynchronous. Control will quickly return to the command prompt, but the force disconnect operation may take multiple seconds or even minutes to complete. This is why the scripts above use `ttStatus` to monitor the status of the force disconnect operation.

The users granularity level includes all connections aside from the subdaemon. For example, in addition to user connections, this includes connections for `ttcserver` and `ttstats`.

Close the database before attempting a forced disconnect process. Any new connection request is rejected by the main daemon during the forced disconnect process. However, after completion of forced disconnect, connection requests are accepted again if the database is not in a closed state.

Set cache policies

Options

ttAdmin has these options for cache:

Option	Description
-connStr *connection_string*	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
DSN	An ODBC data source name of the database to be administered.
-cachePolicy	Defines the policy used to determine when the cache agent for the database should run. manual (default) - Specifies that the cache agent must be manually started and stopped. 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. norestart - Specifies that the cache agent for the database is not to be restarted after a failure.
-cacheStart	Starts a cache agent for the database.
-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.
-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 *Oracle TimesTen Application-Tier Database Cache User's Guide* for more details.
-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. 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.

Option	Description
-cacheUidPwdSet	Sets the cache administration user name and password for the specified database, using the -cacheUid and -cachePwd options. Note the following:
	■ 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 "Making DDL changes in an active standby pair" 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.
-stopTimeout *secs*	Specifies that the TimesTen daemon should stop the cache agent if it does not stop within *secs* seconds.
	If set to 0, the daemon potentially waits forever for the cache agent. The default value is 100 seconds.

Examples

A database referred to by DSN database1 contains data cached from an Oracle database. Use the following ttAdmin command to start the cache agent for database1:

```
% ttAdmin -cacheStart database1
```

You can also use the -cachePolicy option to ask the TimesTen data manager daemon to start the cache agent every time the data manager is started:

```
% ttAdmin -cachePolicy always database1
```

To turn off the automatic start of cache agent:

```
% ttAdmin -cachePolicy manual database1
```

To set the cache administration user ID and password, use -cacheUidPwdSet with -cacheUid and -cachePwd. For example:

```
% ttAdmin -cacheUidPwdSet -cacheUid scott -cachePwd tiger database1
```

To get the current cache administration user ID for database1:

```
% ttAdmin -cacheUidGet database1
```

Notes

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.

When using autorefresh 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 "Managing the cache agent" in *Oracle TimesTen Application-Tier Database Cache User's Guide* for more details.

See also

ttStatus
ttCachePolicySet
ttCacheUidGet
ttCacheUidPwdSet
ttCacheStart
ttCacheStop

Set replication policies

Options

ttAdmin has these options for replication:

Option	Description
-connStr *connection_string*	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
DSN	An ODBC data source name of the database to be administered.
-repPolicy	Defines the policy used to determine when the replication agent starts.
	manual (default) - Specifies that the replication agent must be manually started and stopped.
	always - Specifies that the agent should always be running for the database. This option immediately starts the replication agent. When the daemon restarts, the replication agent is restarted.
	norestart - Specifies that the replication agent for the database is not to be restarted after a failure.
-repQueryThresholdGet	Returns the number of seconds that a query can be executed by the replication agent before TimesTen writes a warning to the daemon log. A value of 0 indicates that no warning is sent.
-repQueryThresholdSet *secs*	This option specifies the number of seconds that a query can be executed by the replication agent before TimesTen writes a warning to the daemon log.
	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.
-repStart	Starts the replication agent.
-repStop	Stops the replication agent.

Examples

These examples show use of replication options.

```
% ttAdmin -repPolicy always rep1
RAM Residence Policy          : inUse
Replication Agent Policy      : always
Cache Agent Policy            : manual
Cache Agent Manually Started  : False
Database State                : Open

% ttAdmin -repPolicy manual rep1
RAM Residence Policy          : inUse
Replication Agent Policy      : manual
Replication Manually Started  : True
Cache Agent Policy            : manual
```

```
Cache Agent Manually Started   : False
Database State                 : Open

% ttAdmin -repPolicy norestart rep1
RAM Residence Policy           : inUse
Replication Agent Policy       : norestart
Replication Manually Started   : True
Cache Agent Policy             : manual
Cache Agent Manually Started   : False
Database State                 : Open

% ttAdmin -repQueryThresholdSet 100 rep1
RAM Residence Policy           : inUse
Replication Agent Policy       : norestart
Replication Manually Started   : True
Cache Agent Policy             : manual
Cache Agent Manually Started   : False
Database State                 : Open

% ttAdmin -repQueryThresholdGet rep1
QueryThreshold in seconds : 100
RAM Residence Policy           : inUse
Replication Agent Policy       : norestart
Replication Manually Started   : True
Cache Agent Policy             : manual
Cache Agent Manually Started   : False
Database State                 : Open
```

Notes

If `ttAdmin` is used with `-repStart` and a replication definition is not found, the
replication agent is not started and `ttAdmin` prints out an error message. For example:

```
% ttAdmin -repstart repl1
*** [TimesTen][TimesTen 18.1.1.0 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 `-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 and Linux 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 patchset or patch release. For example, you can move a database from TimesTen 18.1.1.1.0 to TimesTen 18.1.2.1.0.

> **Note:** A major release refers to the first two parts of the release number (18.1 above). A patchset release refers to the third part of the release number.

This utility is useful for testing a patchset or patch 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.

Usage with TimesTen Scaleout

This utility is not supported in TimesTen Scaleout.

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 (the `info` directory). 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 "Backup, Restore, and Migrate Data in TimesTen Classic" in *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade 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.

Usage with TimesTen Scaleout

This utility is not supported in TimesTen Scaleout.

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 that specifies a database location, driver, and optionally other connection attribute settings.
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 and Linux systems, 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 two parts of the TimesTen release and the platform are the same. For example, you can back up and restore files between TimesTen releases 18.1.1.1.0 and 18.1.2.10. You cannot backup and restore files between releases 11.2.2.8.35 and 18.1.2.1.0. You can use the ttBulkCp or CS (UNIX and Linux only) utility to migrate databases across major releases or operating systems.

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 and Linux systems, 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.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout.

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] [-localOnly] {-connStr connection_string | DSN}
[owner.]tableName [dataFile ...]

ttBulkCp -directLoad [-cp numTrans|final] [-d errLevel] [-e errorFile]
[-m maxErrs] [-sc] [-t errLevel] [-u errLevel]
[-v 0|1] [-xp numRows|rollback] [-Cc | -Cnone]
[-dformat formatStr] [-tformat formatStr]
[-tsformat formatStr] [-F firstRow] [-L lastRow]
[-N ncharEncoding] [-Q 0|1] [-S errLevel] [-dateMode mode]
{DSN | [-connstr] connection_string}
[owner.]tblName [dataFile ...]

ttBulkCp -o [-sc] [-v 0|1] [-A 0|1] [-Cc | -Cnone]
[-nullFormat formatStr] [-localOnly]
[-tformat timeFormat] [-tsformat timeStampFormat]
[-dateMode dateMode] [-dformat | -D dateFormat]
[-N ncharEncoding] [-noForceSerializable | -forceSerializable]
```

```
[-tsprec precision] [-Q 0|1] [-localOnly]
{-connStr connection_string | DSN} [owner.]tblName
[dataFile]
```

Options

ttBulkCp has the options:

Option	Description
-Cnone -Cc	-Cnone disables the use of comments in the output file. -Cc sets the default comment character to c. If no default comment character is specified, the pound character (#) is used. The -C option takes the values: \t (tab) or any of the characters: ~ ! @ # % ^ & * ( ) = : ; | < > ? , / This option overrides the COMMENTCHAR file attribute.
-connStr connection_string	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
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 5-32. 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.
-directLoad	Selects copy-in mode that copies data from an ASCII file into a database table, but can only be used by a client using a direct connection. Avoids some of the overhead required when using a client/server connection, which provides better performance than the -i mode. Can only be used with TimesTen Classic.
-h -help -?	Prints a short usage message and exits.
-helpfull	Prints a longer usage message and exits.
-i	Selects copy-in mode that copies data from an ASCII file into a database table. Can be used by a client using either a direct connection or a client/server connection.

Option	Description
-localonly	This option only loads rows from a specific instance. Load a specific instance in the grid and use this option. When you use this option, ttBulkCP selects all rows from the table, but ignores any rows that are not hashed to the specific instance. This option is only supported in TimesTen Scaleout. The default value is N.
-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."
-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 5-32. 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 5-32. 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*.
-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) and directload (-directload) modes 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.
	NOTE: This option is not supported in TimesTen Scaleout.

Option	Description
-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 5-28), 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`.
-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.

Option	Description
-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.
-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 5-29). 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 5-29 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 5-30 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 5-32. 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 5-32. 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 5-32. 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.

- \uxyzw (NCHAR and NVARCHAR2 only) The character whose unicode value is xyzw, where xyzw is a four-digit hexadecimal number, as in \ufe4a. The \uxyzw notation is supported in both UTF-8 and ASCII encoding modes.

In addition, any of the ~ ! @ # $ % ^ & * ( ) = : ; | < > ? , / characters can be escaped with a backslash. Although it is unnecessary to escape these characters 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 .FFn.
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
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 2-46.

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 in 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.

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

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.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout.

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.

Option	Description
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.
-h	Prints a usage message and exits.
-help	
-?	
-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 and Linux 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.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout.

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 that specifies a database location, driver, and optionally other connection attribute settings.
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.logn, *dspath* is /home/payroll/2011.
-f *outFile*	Specifies the output file name; defaults to stdout.
-h -help -?	Prints a usage message and exits.

Option	Description
-header	Checks the content of the database header.
-heap	Checks all heap structures.
-indexHeap	Checks the index heap structure.
-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, see "Using Oracle Clusterware to Manage Active Standby Pairs" in *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, any user with Administrators privileges can execute all commands in this utility.

On UNIX and Linux systems, 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.

Usage with TimesTen Scaleout

This utility is not supported in TimesTen Scaleout.

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 - [-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 [-noderegister] [-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 and Linux 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 and Linux systems, only the root user can execute this command.

Option	Description
-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. See also: -endAlterSchema.
-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 and Linux 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 and Linux 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

Option	Description
-h -help -?	Prints a usage message and exits.
-init	Starts the TimesTen cluster agent.
-noderegister	Used with `ttCWAdmin -shutdown` command. This command option tells the shutdown process to keep registered all TimesTen processes that are registered as Clusterware resources for the cluster agent and TimesTen daemon monitors for Clusterware.
-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.

Option	Description
-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 daemon, cluster agent, and the replication agent (if the replication agent is still up) on the set of hosts either specifically mentioned with the optional -hosts argument or defined within the ttcrsagent.options file. Also, if this command does not include the -noderegister option, then the default behavior is to deregister from Clusterware all TimesTen processes that are registered as Clusterware resources for cluster agents and TimesTen daemon monitors.
-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.
-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.
-	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 - 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.

- 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.

Usage with TimesTen Scaleout

This utility is not supported in TimesTen Scaleout.

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 timesten.conf 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, 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 `timesten.conf` 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 `timesten.conf` options:

```
server_port server_shmipc server_shmsize noserverlog
```

See also

For a description of all daemon options and instructions for changing the `timesten.conf` file, see Chapter 1, "TimesTen Instance Configuration File" in this reference and "Managing TimesTen daemon attributes" in the *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 daemon 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.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout.

Syntax

```
ttDaemonLog {-h | -help | -?}
ttDaemonLog {-V | -version}
ttDaemonLog [-show type] [-b | -r | -s] [-f] [-maxlines]
[-loglevel level [DSN | -connstr connection_string]]
[-[no]logcomponent component [DSN | -connstr connection_string]]
[-logreset] [-msg messagestring] [-setquiet | -setverbose]
[-file filename] [-facility name]
[-n computer]
```

> **Notes:**
>
> - The -file and -facility options apply only on UNIX and Linux.
>
> - 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.

Option	Description
-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.and Linux.
-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 and Linux.
-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 GADMIN: For messages from grid administrator GADMINCS: For messages from grid administrator client server GCWRKR: For messages from grid client worker GRID: For messages from a TimesTen Grid GRWKR: For messages from the grid worker REPLICATION: For messages from the replication agent, designated by REP TTSTATS: For messages from ttStats
-loglevel *level*	Specifies a cutoff for the level of messages that are logged in the daemon 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.

Option	Description
-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.
-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
```

On UNIX and Linux systems, 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 `timesten.conf` file. See "Error, warning, and 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.

Usage with TimesTen Scaleout

This utility is not supported in TimesTen Scaleout.

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 that specifies a database location, driver, and optionally other connection attribute settings.
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.logn, *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.

`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.

ttInstallationCheck

Description

The ttInstallationCheck utility examines all files in an installation of TimesTen and will generate a signature for the installation. The signatures from two installations can be compared; if there are any differences in the installations the signatures differ.

If any of the following have occurred, the signature reported is different:

- Contents of any file have changed
- Name of any file has changed
- New files are present in the installation
- Files have been removed from the installation
- Files have incorrect permissions

Required privilege

This utility requires the instance administrator privilege.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout.

Syntax

```
ttInstallationCheck [-h | -help | -?]
ttInstallationCheck [-v | -verbose | -?]

ttInstallationCheck [-install_dir path] [-generate]
```

Options

ttInstallationCheck has the options:

Option	Description
-h -help -?	Displays help information.
-generate	Generate and print the checksum for the installation, but do not verify.
-install_dir path	Specifies the directory in which TimesTen is installed.
-v	-verbose	Displays extra installation information.

ttInstallDSN

Description

The ttInstallDSN utility, for TimesTen Scaleout, generates a Windows client DSN for each of one or more entries in the provided input file and installs them into the ODBC control panel as system DSNs. Use the ttGridAdmin gridClientExport command to generate the input file.

Required privilege

This utility requires the instance administrator privilege.

Usage with TimesTen Scaleout

This utility is for use with TimesTen Scaleout.

Syntax

```
ttInstallDSN [-h | -help | -?]

ttInstallDSN [-f file] [Client_DSN_Name | -a | -l] [-force]
```

Options

ttInstallDSN has the options:

Option	Description
-h -help -?	Displays help information.
-f *file* [*Client_DSN_Name*	-a	-l]	Specifies the full path and name of a file (generated by ttGridAdmin gridClientExport) containing one or more DSN definitions. Also typically provide one of the following: ■ The name of the DSN you want to install from the input file. ■ The -a option to install all DSNs from the input file. ■ The -l option to simply list all the DSNs in the input file. None is installed. If you do not specify -f, the default file is sys.odbc.ini in the current directory. If you do not specify one of the three items listed above, the default behavior is to list the DSNs in the file and ask which you want to install.
-force	If there are already DSNs in the Windows registry, this allows them to be overridden by the specified DSNs. Without this option, if there are already DSNs in the Windows registry, the utility issues a warning and cannot install new ones.

Examples

In this example, there are already DSNs in the Windows registry. The user first tries without -force and is issued a warning, so then uses -force.

```
C:\mydir> ttinstalldsn.bat -f c:\temp\sys.odbc.ini
```

```
--------------------------------
.ini File: c:\temp\sys.odbc.ini
--------------------------------
Found the following DSNs in available 'c:\temp\sys.odbc.ini'.
0 : database1CS
[ Please select the DSN to be imported: ]
0
Warning: The following DSNs already existed and were not added:
        database1CS

C:\mydir> ttinstalldsn.bat -f c:\temp\sys.odbc.ini -force
--------------------------------
.ini File: c:\temp\sys.odbc.ini
--------------------------------
Found the following DSNs in available 'c:\temp\sys.odbc.ini'.
0 : database1CS
[ Please select the DSN to be imported: ]
0
Modifying DSN 'database1CS'.
```

ttInstanceCreate

Description

The `ttInstanceCreate` utility creates a new TimesTen instance.

You can specify options in one of these ways:

- On the command line.

- In a file.

- Interactively as the utility runs.

If you do not specify options on the command line, or if the only options used are `-record` and/or `-verbose`, `ttInstanceCreate` runs in an interactive mode, prompting the Instance Administrator for information.

If you specify the `-batch` option on the command line, `ttInstanceCreate` runs in interactive mode, and attempts to answer any questions by fetching the answers from a recorded batch file, generated by a previous run that specified the `-record` option. If the answer to a question is not present in the batch file, the utility prompts the Instance Administrator to answer the question interactively.

If you specify other options on the command line, they are used as the source of information. The `ttInstanceCreate` utility does not prompt the user for unknown values.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout but is used only to create the first management instance. (Create additional instances using `ttGridAdmin instanceCreate`.)

Required privilege

This utility requires the instance administrator privilege.

Syntax

```
ttInstanceCreate {-h | -help | -?} [-verbose]
```

To create an instance for TimesTen Classic, use:

```
ttInstanceCreate [-name name] [-location path] [-daemonport port] [-csport port]
 [-start] [-tnsadmin path] [-force] [-record filename] [-strict] [-verbose]
```

To create the first management instance for a grid in TimesTen Scaleout, use:

```
ttInstanceCreate [-name name] [-location path] [-daemonport port] [-csport port]
 [-grid] [-force] [-record filename] [-strict] [-verbose]
```

To create a client-only instance, use:

```
ttInstanceCreate [-name name] [-location path] [-clientonly] [-serverhost host]
 [-force] [-record filename] [-strict] [-verbose]

ttInstanceCreate [-batch [filename]]
```

Options

ttInstanceCreate has the options:

Option	Description
-h -help -?	Displays help information.
-name *name*	Specifies the name of the instance to be created.
-location *path*	Specifies the path of the directory in which the instance is to be created. This directory must already exist. A new directory is created in the existing directory. The name of the new directory is specified in the -name option. This new directory is the new instance home.
-force	Specifies that an instance directory specified with the -instance option is to be overwritten if it already exists. The directory is overwritten only if: 1. The specified instance directory is empty, or 2. The specified instance directory contains a conf/timesten.conf file.
-record *filename*	Records responses to installation questions into the file specified by *filename*. The file then can be specified as the parameter to the -batch option.
-batch [*filename*]	Specifies the file to be used to provide input to the ttInstanceCreate utility. If not specified, no input file is used.
-strict	Ensures that the platform running the command is supported and prevents the instance creation if it is not.
-tnsadmin *location*	If using Cache, this option configures the location to be used for the TNS_ADMIN setting.
-verbose	Displays additional information during the operation of the utility.
-grid	Indicates that the instance should be configured for use with TimesTen Scaleout.
-clientonly	Specifies that the instance is client only. Most other arguments are not supported for a client only instance.

Use these options for full instances with client and server capabilities:

Option	Description
-daemonport *daemon_port*	The port number on which the TimesTen daemon process (timestend) for this instance listens. This port must not already be in use by any other application or instance on the system.
-csport *port*	The value to be used for the TimesTen client/server port number for this instance. If not specified, the default is *daemonport* + 1.

Use this option for instances intended for TimesTen Classic:

Option	Description
-start	Specifies that the instance should be started after it is created.

Use this option for client-only instances:

Option	Description
-serverhost *host*	For TimesTen Client, specify the host for the TimesTen Server. The value is the logical name, domain name, or IP address of the computer where the TimesTen Server is running. For a full instance, this defaults to the host where you are creating the instance.

ttInstanceDestroy

Description

Use the ttInstanceDestroy utility to destroy an existing instance.

The instance to be destroyed is chosen based on the current setting of the TIMESTEN_ HOME environment variable.

Required privilege

This utility requires the instance administrator privilege.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout., but in most circumstances use ttGridAdmin instanceDelete.

Syntax

```
ttInstanceDestroy {-h | -help | -?} [-verbose]

ttInstanceDestroy [-force]
```

Options

ttInstanceDestroy has the options:

Option	Description
-h -help -?	Displays help information.
-force	If specified, you are not asked to confirm operations. If -force is not specified: ■ You are reminded that if you installed the startup scripts for this instance as root, you must uninstall them as root with the setuproot -uninstall command. ■ You are asked for confirmation to destroy the instance. ■ You are asked for confirmation to remove the info and conf directories.
-verbose	Displays additional TimesTen installation information.

ttInstanceModify

Description

Use the `ttInstanceModify` utility to modify certain attributes of an instance, including:

- The installation associated with this instance.

- The daemon and server port numbers.

- The `TNS_ADMIN` for the instance

- The configuration of TimesTen Replication with Oracle Clusterware for this instance.

The instance that is modified is the one that `$TIMESTEN_HOME` references.

If you do not specify any options for this utility, `ttInstanceModify` displays the current value of each attribute and a prompt that allows you to keep the value or change it.

If you change any of the settings, the utility:

- Shuts down the TimesTen daemon for the instance.

- Edits the `timesten.conf` file in the *timesten_home*`/conf` directory.

- Starts the TimesTen main daemon for the instance.

Required privilege

This utility requires the instance administrator privilege.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout, but in most circumstances use `ttGridAdmin instanceModify`.

Syntax

```
ttInstanceModify [-h | -help | -?] [-verbose]

ttInstanceModify [-port daemonport] [-serverport cs_port] [-tnsadmin location]
[-crs] [-install installation_dir]
```

Options

`ttInstanceModify` has the options:

Option	Description
-h -help -?	Displays help information.
-daemonport *daemonport*	Updates the TimesTen main daemon port number.
-crs	Creates or modifies the instance's Oracle Clusterware configuration.

Option	Description
-install *installation_dir*	Changes the installation that the instance uses.
	You can use the -install option to associate the instance with a different TimesTen installation. Typically, this is used to upgrade the instance to a new maintenance or patch release.
	This option cannot be used to upgrade to a new major release, for example to upgrade from 11.2.2. to 18.1.
-tnsadmin *location*	Updates the instance's TNS_ADMIN setting (for cache).

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 and Linux systems, 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.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout.

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 that specifies a database location, driver, and optionally other connection attribute settings.
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.

Option	Description
-interactive	Forces interactive mode. This is useful when running from an emacs comint buffer.
-N ncharEncoding	Specifies the character encoding method for NCHAR output.
	Valid values are LOCALE or ASCII. LOCALE (the default) sets the output format to the locale-based setting.
	If no value is specified, TimesTen uses the system's native language characters.
-V | -version	Prints the release number of ttIsql and exits.
-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 5-83.

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.
`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.

Command	Description
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.
builtins [*builtin_name_ pattern*]	Lists, in a single column, the names of all the TimesTen built-in procedures that match the given pattern. When the pattern is missing, the pattern defaults to "%".
	See the procedures command.
bye exit	Exits ttIsql.
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.
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.

Command	Description
cd *directory*	Changes the current directory.
	This is the equivalent of the cd command in interactive shells.
	After changing to the directory *directory*, the define alias _CWD is be set to this directory.
	Subsequent commands that rely on relative paths will use this directory as the starting point.
	Examples of affected commands are spool, run, savehistory, host, and edit
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.
	See "Choose SQL and PL/SQL timeout values" in *Oracle TimesTen In-Memory Database Operations Guide* for information about the relationship between the client timeout, SQL timeout, and PL/SQL timeout.
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.
	Use prepare to create the prepared command.
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 Durability=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 5-83.

Command	Description
`describe [[`*`owner_pattern`*`.]` *`name_pattern`* `|` *`procedure_name_ pattern`* `|`*`sql_statement`* `|` `[`*`connect_id`*`.]`*`command_id`* `|*]`	List information on tables, synonyms, views, materialized views, 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 or materialized view being described is in a TimesTen Scaleout database, this command reports the distribution scheme.
	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.
	If the command is describing a sequence in a TimesTen Scaleout database, displays the batch field.
	The command alias is `desc`.
	Use `free` to release the prepared command.
`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 size information in KB, MB, GB or TB. For TimesTen Scaleout, provides the size of the element.
	The default is MB. 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 Linux systems 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.
	Use free to release the prepared 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.
	Use free to release the prepared 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.
	Use free to release the prepared command.
fetchnext *num_rows* [*connect_id.*]*command_id*]	Fetches up to *num_rows* rows from prepared command *command_id* on connection *connect_id*.
	If *command_id* is not specified, fetches *num_rows* rows from the most recent command. The command must already have been executed using exec.
	Use free to release the prepared command.
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.
	Use free to release the prepared command.
free [[*connect_name.*]*connect_id.*] *command_id*]	Frees prepared command *command_id* on connection *connect_id*.
	If no command is specified, frees the most recent command.
	Use prepare to create the prepared 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.
grid stmt	Performs that specified statement on a grid database.
grid monitor [*optional_monitor_column*]	Formats the contents of the SYS.GV$MONITOR table for easy viewing.
	If the *optional_monitor_column* is specified, only that column is displayed.
	This command is not supported in TimesTen Classic.

Command	Description
help [*command* [*command* ...]| all | comments | attributes]	Prints brief or detailed help information for commands.
	If specific commands are given as arguments then detailed help for each command is printed.
	If you 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.
history [-all] [-h] [-r] [*num_commands*]	ttIsql implements a csh-like command history.
	Lists previously executed commands. The *num_commands* parameter specifies the number of commands to list. If the *num_commands* parameter is omitted then the previous 10 commands are listed by default.
	The output of this command omits consecutive duplicate commands. Use the -all option to include the consecutive duplicate commands.
	Use the -h option to omit the command numbers.
	Use the -r parameter to list the commands in reverse order.
	The history list stores up to 100 of the most recently executed commands.
	See the savehistory and clearhistory 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 5-80.
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.

Command	Description
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.
	Use free to release the prepared command.
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.
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.

Command	Description
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.
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 5-92 for a description of *arguments*.
savehistory [-all [-h] [-a | -f] *outputfile*	Writes the history buffer to the specified *outputfile*. Consecutive duplicate commands are omitted.
	Use the -all option to include the consecutive duplicate commands.
	Use the -h option to omit the command numbers.
	Use -a to append to an existing output file. Use -f to force the overwriting of an existing 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 5-83.
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.

Command	Description
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*.
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 5-83.
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.
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.
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.

Command	Description
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 5-81.
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.
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 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 5-67. 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 5-95 for an explanation of the default.
dynamicloadenable [1|0]	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.

Attribute	Description
editline [0 | 1]	Turns the editline function off and on. By default, editline is on.
	If editline is turned off, the backspace character deletes full characters, but the rest of editline capabilities are unavailable.
err | error |errors [*objecttype* [*schema.*]*name*]	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 multiple connection 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.

Attribute	Description
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.
nulldisplaystring "*string*"	Sets or shows the string to be displayed when the NULL value appears in a result set.
	The option does not affect the SQL user, only the display of NULL in results sets.
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]	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.
	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.

Attribute	Description
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.
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 daemon log.
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.
	This command is not supported in passthrough mode.
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.

Attribute	Description
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.
	See "Choose SQL and PL/SQL timeout values" in *Oracle TimesTen In-Memory Database Operations Guide* for information about the relationship between the client timeout, SQL timeout, and PL/SQL timeout.
timing [1|0]	Enables or disables printing of query timing.
	You can also enable this attribute without specifying the set command.
tryhash [1|0]	Enables or disables use of hash indexes by 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.

Attribute	Description
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.
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] */
```

The C-style comments, delineated by "/*" at the beginning and "*/" at the end, can span multiple lines.

The comments delimited by the

-

character 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 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.
	If the command being added to the history is identical to the most recently added command, it is skipped.
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.
	If the command being added to the history is identical to the most recently added command, it is skipped.
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: `?`

Examples

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;
        exec :a := 110;

PL/SQL procedure successfully completed.

Command> print a
A                   : 110
Command> var b number;
        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.
```

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=/sw/tthome/install/lib/libtten.so; PermSize=20;TempSize=20;
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 -connStr "DSN=RunData"
ttIsql (c) 1996-2011, TimesTen, Inc. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
(Default setting AutoCommit=1)
Command> prepare 1 SELECT * FROM my_table;
        exec 1;
        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;
```

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 `xlaSimple`, enter:

```
Command> xlabookmarkdelete xlaSimple;
```

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
        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
        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.
```

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;
         VARIABLE name VARCHAR2(25) INLINE NOT NULL;
         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.

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
         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 = ?;
         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
- Sequences
- Replication schemes
- Users and user information

Use the `ttMigrate` utility when upgrading major release versions of TimesTen, since database checkpoint and log files are not compatible between major releases. See "Moving to a different major release of TimesTen Classic" in *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide*.

When you migrate a database from Release 7.0 or earlier, TimesTen does not migrate 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 and Linux systems 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 and Linux systems, 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.

Usage with TimesTen Scaleout

This utility is supported for migrating from a TimesTen Classic to a TimesTen Scaleout. After the initial migration, this utility is not supported.

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]
[-activeDML | -noActiveDML]
{DSN | -connStr connection_string} 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] [-restorePublicPrivs] [-localhost host]
[-resizeHashIndexes]
{DSN | -connstr connection_string} 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.]objectName...]
```

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 5-103 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 5-103 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 that specifies a database location, driver, and optionally other connection attribute settings.
-d	Selects Describe mode. Displays a short description of the objects in the data file. See "Describe mode (-d)" on page 5-106 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 5-107 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.
-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.

Option	Description
-inline *rule*	Indicates the rule to be used for converting variable-length columns to INLINE in restore mode. The value for rule is one of: preserve - ttMigrate preserves the original INLINE attribute of each column. This is the default, and it is required if -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 5-106 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 5-106 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 5-105 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.
-relaxedUpgrade	Save or restore the tables in a way that is compatible with a replication scheme that uses TABLE DEFINITION CHECKING RELAXED. 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.

Option	Description
-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 *ckptFreq*	Specifies that ttMigrate should checkpoint the database after restoring every *ckptFreq* megabytes of data. A value of zero (the default) specifies that ttMigrate should never checkpoint the database.
	NOTE: This option is not supported in TimesTen Scaleout.

Option	Description
-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.
-gridRestoreFinale	Restores indexes and foreign keys. Use this option only for TimesTen Classic to TimesTen Scaleout migration. See "Migrating a database from TimesTen Classic to TimesTen Scaleout" in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*.
-gridRestoreRows	Restores rows into tables. Use this option only for TimesTen Classic to TimesTen Scaleout migration. See "Migrating a database from TimesTen Classic to TimesTen Scaleout" in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*.
-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-18.1 data into a 18.1 or later release of TimesTen 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.
-resizeHashIndexes	Resizes user hash indexes during restore to be optimal size based on number of table rows.
-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.

Cache group data type conversions

When restoring a database that contains cache groups from a TimesTen release that is earlier than 7.0, use the -convertCGTypes. option to convert the data type of columns from pre-7.0 types to more clearly map with the data types of the columns in the Oracle database with which the cache group is associated.

The following table describes the type mapping.

Pre-7.0 TimesTen Type	Oracle Type	Converted Type
TINYINT	NUMBER(p,s) when s > 0	NUMBER(p,s)
TINYINT	NUMBER(p,s) when s <= 0	TT_TINYINT
SMALLINT	NUMBER(p,s) when s > 0	NUMBER(p,s)
		TT_SMALLINT
SMALLINT	NUMBER(p,s) when s <= 0	TT_SMALLINT
INTEGER	NUMBER(p,s) when s > 0	NUMBER(p,s)
INTEGER	NUMBER(p,s) when s <= 0	TT_INTEGER
BIGINT	NUMBER(p,s) when s > 0	NUMBER(p,s)
BIGINT	NUMBER(p,s) when s <= 0	TT_BIGINT
NUMERIC(p,s)DECIMAL(p,s)	NUMBER	NUMBER
NUMERIC(p,s)DECIMAL(p,s)	NUMBER(x,y)	NUMBER(x,y)
NUMERIC(p,s)DECIMAL(p,s)	FLOAT(x)	NUMBER(p,s)
REAL	Any	BINARY_FLOAT
DOUBLE	Any	BINARY_DOUBLE
FLOAT(x) x <=24	Any	BINARY_FLOAT
FLOAT(x) x >= 24	Any	BINARY_DOUBLE
CHAR(x)	Any	ORA_CHAR(x)
VARCHAR(x)	Any	ORAVARCHAR2(x)

Pre-7.0 TimesTen Type	Oracle Type	Converted Type
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`.

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.

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.

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.

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`.

- 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 and Linux systems, 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

ttRepAdmin

Description

Displays existing replication definitions and monitors replication status. The ttRepAdmin utility is also used when upgrading to a new release of TimesTen

Required privilege

This utility requires the ADMIN privilege.

Usage with TimesTen Scaleout

This utility is not supported in TimesTen Scaleout.

Syntax

```
ttRepAdmin {-h | -help | -?}
ttRepadmin {-V | -version}
ttRepAdmin -self -list [-scheme [owner.]schemeName]
        {DSN | -connStr connection_string}

ttRepAdmin -receiver [-name receiverName]
        [-host receiverHostName] [-state receiverState] [-reset]
        [-list] [-scheme [owner.]schemeName]
        {DSN | -connStr connection_string}

ttRepAdmin -log {DSN | -connStr connection_string}

ttRepAdmin -showstatus -detail {-awtmoninfo} {DSN | -connStr connection_string}

ttRepAdmin -showconfig {DSN | -connStr connection_string}

ttRepAdmin -bookmark {DSN | -connStr connection_string}

ttRepAdmin -wait [-name receiverName] [-host receiverHostName]
        [-timeout seconds] {DSN | -connStr connection_string}

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]
        [ (  -activeDataGuard [-cacheUid cacheUid [-cachePwd cachePwd]]
           | -initCacheDr [-cacheUid cacheUid [-cachePwd cachePwd]]
                   [-noDRTruncate] [-nThreads]
           | ( -keepCG [-cacheUid cacheUid [-cachePwd cachePwd]]
                ( [-recoveringNode | -deferCacheUpdate] ))| -nokeepCG ) ]
        [-remoteDaemonPort portNo] [-verbosity {0|1|2}]
        [-localhost localHostName]
        [-open | -close]
        {destDSN | -connStr connection_string}
```

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	Display help information.
-help	
-?	
-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 connection_string}
```

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, an ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
-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 connection_string}
```

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, an ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
-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.
-state stop	start (default) - Starts replication to the subscriber.
-state pause	stop - Stops replication to the subscriber, discarding updates.
	pause - Pauses the replication agent, preserving updates.
	See "Set the replication state of subscribers" in *Oracle TimesTen In-Memory Database Replication Guide* for more information.
-reset	Clears the bookmark in the master database log for the latest transaction to be sent to a given subscriber. This option should only be used when the transaction numbering of the master database is changed, such as when the database is re-created using ttMigrate or ttBackup. If the master database is saved and restored using ttBackup and ttRestore, transaction numbering is preserved and this option should not be used.
-list	Lists information about a replication definition.
-scheme [*owner.*]*schemeName*	Specifies the replication scheme name when there is more than one scheme.

Examples

```
% ttRepAdmin -receiver -list my_dsn
```

The above syntax lists replication information for all the subscribers of the master database, my_dsn.

```
% ttRepAdmin -receiver -name rep_dsn -list my_dsn
```

The above syntax lists replication information for the rep_dsn subscriber of the master database, my_dsn.

```
% ttRepAdmin -receiver -name rep_dsn -reset my_dsn
```

The above syntax resets the replication bookmark with respect to the rep_dsn subscriber of the master database. Should only be used when migrating a replicated database with ttMigrate or ttBulkCp.

```
% ttRepAdmin -receiver -name rep_dsn -state Start my_dsn
```

The above syntax resets the replication state of the rep_dsn subscriber database to the Start state with respect to the master database, my_dsn.

Duplicate a database

Use this form of `ttRepAdmin` to create a new database with the same contents as the master database.

The following must be true for you to perform the `ttRepAdmin -duplicate`:

- Only the instance administrator can run `ttRepAdmin -duplicate`.

- The instance administrator must have the same operating system username on both source and target computer to execute `ttRepAdmin -duplicate`.

- You must provide the user name and password with the `-UID` and `-PWD` options for 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]
    [ ( -activeDataGuard [-cacheUid cacheUid [-cachePwd cachePwd]]
       | -initCacheDr [-cacheUid cacheUid [-cachePwd cachePwd]]
             [-noDRTruncate] [-nThreads]
      | ( -keepCG [-cacheUid cacheUid [-cachePwd cachePwd]]
         ( [-recoveringNode | -deferCacheUpdate ] ))| -nokeepCG  ) ]
    [-remoteDaemonPort portNo] [-verbosity {0|1|2}]
    [-localhost localHostName]
    [-open | -close]
    {destDSN | -connStr connection_string}
```

Options

`ttRepAdmin -duplicate` has the options:

Option	Description
-close	Closes a database to user connections.
	When a database is closed to user connections, new connection attempts will fail, but existing connections are unaffected.
-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, an ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.

Option	Description
-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.
-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 [-cacheUid *cacheUid* -cachePwd *cachePwd*]	Initializes disaster recovery. Must provide the cache admin user id and password. ■ *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.
(-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.

Option	Description
-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.
-open	Opens a database to user connections. A database is open to user connections by default upon creation.
-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.
-ramLoad	Keeps the database in memory upon completion of the duplicate operation. This option avoids the /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 5–1   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 5–2   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 5–3   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 connection_string ]
```

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 connection_string}
```

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, an ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
-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 connection_string}
ttRepAdmin -showstatus -detail {-awtmoninfo} {DSN | -connStr connection_string}
ttRepAdmin -showconfig {DSN | -connStr connection_string}
ttRepAdmin -bookmark {DSN | -connStr connection_string}
```

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 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, an ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
-detail	Indicates whether the replication agent transmitters and receivers are using TLS (indicated as SSL)
-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 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, Oracle execution and Oracle commits.

See "Use ttRepAdmin to show replication status" in *Oracle TimesTen In-Memory Database Replication Guide* for more information.

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.

For example, consider how to use the `ttRepAdmin -showstatus` utility to display status for a unidirectional replication scheme from the `rep1` database to the `rep2` database.

The first `ttRepAdmin -showstatus` output shows the status of the `rep1` database and its `TRANSMITTER` thread. The second output shows the status of the `rep2` database and its `RECEIVER` thread.

```
% ttRepAdmin -showstatus rep1

DSN                  : rep1
Process ID           : 1980
Replication Agent Policy : MANUAL
Host                 : MYHOST
RepListener Port     : 1113 (AUTO)
Last write LSN       : 0.1487928
Last LSN forced to disk  : 0.1487928
Replication hold LSN : 0.1486640

Replication Peers:
  Name               : rep2
  Host               : MYHOST
  Port               : 1154 (AUTO)
  Replication State  : STARTED
  Communication Protocol : 12

TRANSMITTER thread(s):
  For                : rep2
    Start/Restart count  : 2
    Send LSN           : 0.1485960
    Transactions sent  : 3
    Total packets sent : 10
    Tick packets sent  : 3
    MIN sent packet size : 48
    MAX sent packet size : 460
    AVG sent packet size : 167
    Last packet sent at  : 17:41:05
    Total Packets received: 9
    MIN rcvd packet size : 48
    MAX rcvd packet size : 68
    AVG rcvd packet size : 59
    Last packet rcvd'd at : 17:41:05
    Earlier errors (max 5):
    TT16060 in transmitter.c (line 3590) at 17:40:41 on 08-25-2004
    TT16122 in transmitter.c (line 2424) at 17:40:41 on 08-25-2004
```

The replication status for the rep2 database should look similar to the following:

```
> ttRepAdmin -showstatus rep2

DSN                  : rep2
Process ID           : 2192
Replication Agent Policy : MANUAL
Host                 : MYHOST
RepListener Port     : 1154 (AUTO)
Last write LSN       : 0.416464
Last LSN forced to disk  : 0.416464
Replication hold LSN : -1.-1

Replication Peers:
  Name             : rep1
  Host             : MYHOST
  Port             : 0 (AUTO)
  Replication State : STARTED
  Communication Protocol : 12

RECEIVER thread(s):
  For              : rep1
  Start/Restart count  : 1
```

```
      Transactions received : 0
      Total packets sent    : 20
      Tick packets sent     : 0
      MIN sent packet size  : 48
      MAX sent packet size  : 68
      AVG sent packet size  : 66
      Last packet sent at   : 17:49:51
      Total Packets received: 20
      MIN rcvd packet size  : 48
      MAX rcvd packet size  : 125
      AVG rcvd packet size  : 52
      Last packet rcvd'd at : 17:49:51
```

```
% 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 database1

[other -showstatus output]
...
AWT Monitoring statistics
-------------------------
TimesTen processing time : 0.689000 millisecs (0.164307 %)
   Oracle bookmark  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 "Upgrades in TimesTen Classic" in *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade 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 "Backup, Restore, and Migrate Data in TimesTen Classic" in the *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide*.

Required privilege

This utility requires the instance administrator privilege.

Usage with TimesTen Scaleout

This utility is not supported in TimesTen Scaleout.

Syntax

```
ttRestore {-h | -help | -?}
ttRestore {-V | -version}
ttRestore [-fname filePrefix] [-noconn] -dir directory
        [-open | -close] {DSN | -connStr connection_string}
ttRestore -i [-noconn] [-open | -close] {DSN | -connStr connection_String}
```

Options

ttRestore has the options:

Option	Description
-connStr connection_string	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
-close	Closes a database to user connections.
	When a database is closed to user connections, new connection attempts will fail, but existing connections are unaffected.
DSN	Specifies an ODBC data source name of the database to be administered.
-dir directory	Specifies the directory where the backup files are stored.

Option	Description
-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.
-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 ed from memory.
-open	Opens a database to user connections.
	A database is open to user connections by default upon creation.
-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 and Linux systems, 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 back up and restore databases only when the first two parts of the TimesTen release and the platform are the same. For example, you can back up and restore files between release 18.1.1.1.0 and release 18.1.1.2.0 or 18.1.2.1.0. You cannot back up and restore files between release 11.2.2.8.0 and release 18.1.1.1.0, or between release 18.1.2.1.0 and release 18.2.1.1.0. You can use the ttBulkCp or ttMigrateCS (UNIX and Linux systems only) utility to migrate databases across major releases or operating systems.

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
- Column definitions, including partition information
- PL/SQL program units
- Users and user information

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 and Linux systems, 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.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout.

Syntax

```
ttSchema {-h | -help | -?}
ttSchema {-V | -version}
ttSchema [-userinfo] [-includePassword]
ttSchema [-l] [-c] [-fixedTypes] [-st | -systemTables]
        [ -list {all | tables | views | sequences |
        cachegroups | repschemes |synonyms | plsql | userinfo} [,...] ]
        [-plsqlAttrs | -noplsqlAttrs]
        [-plsqlCreate |-[no]plsqlCreateOrReplace]
        [-includePassword]
        {-connStr connection_string | DSN }
        [[owner.]object_name][...]
```

Options

ttSchema has the options:

Option	Description
-connStr *connection_string*	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
-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.
-h -help -?	Prints a usage message and exits.
-includePassword	Includes exportable password information for users. When the user DDL is executed in a new database, the user can have the same password.
-l	One per-line listing of objects in the database.
-list {all | tables | views | sequences | cachegroups | repschemes |synonyms | plsql |userinfo}[,...]	A comma-delimited (no space after comma) list of objects to generate. Lists only those types of objects specified. Default is -list all.
[*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 5–4   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 5–5   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 5–6   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 5–7   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.

Usage with TimesTen Scaleout

This utility is supported in TimesTen Scaleout.

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 that specifies a database location, driver, and optionally other connection attribute settings.
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. tableName is an optional parameter. The output of ttSize is not very useful if you omit the tableName parameter.
-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

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

The ttStats utility is supported in TimesTen Classic and TimesTen Scaleout, but supports different syntax and options. The following sections describe the ttStats utility for:

- TimesTen Classic

- TimesTen Scaleout

Table 5–3 shows the main differences of how the ttStats utility works in TimesTen Classic and TimesTen Scaleout.

Table 5–3 Differences between ttStats in TimesTen Classic and TimesTen Scaleout

Action	TimesTen Classic	TimesTen Scaleout
Create a snapshot	Execute the ttStats -snapshot utility to create a snapshot. The -description option for -snapshot is optional	The ttStats daemon creates snapshots based on the pollSec parameter of the ttStatsConfig built-in procedure. If you execute ttStats -snapshot -description *description* in TimesTen Scaleout, you can associate a description to the latest system generated snapshot.
Drop a snapshot	Execute the ttStats -drop -begin_snap *snapid1* -end_snap *snapid2* utility to drop snapshots.	The ttStats daemon automatically drops snapshots based on the retentionDays parameter of the ttStatsConfig built-in procedure. The -drop option of the ttStats utility is not supported in TimesTen Scaleout.
View information about a snapshot	Execute the ttStats -report utility without any arguments. The ttStats utility prints the snapshot ID, date, time, capture level, and the description of all snapshots.	Execute the ttStats -snapshotInfo utility. The ttStats utility prints the snapshot ID, date, time, and the description of all snapshots.
Generate a report between two snapshots	Execute the ttStats -report -snap1 *snapid1* -snap2 *snapid2* utility. The utility generates a report between two snapshot IDs.	Execute the ttStats -report -snap1 *snapid1* -snap2 *snapid2* utility or the ttStats -report -timestamp1 *ts1* -timestamp2 *ts2* utility. The utility generates a report between two snapshot IDs or timestamps. You can only generate a text version of a report in TimesTen Scaleout.

TimesTen Classic

This section describes how the `ttStats` utility works in TimesTen Classic.

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.)

The `ttStats` utility gathers metrics from TimesTen Classic 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, 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 and Linux systems 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.

For information about built-in procedures mentioned, and the data they gather, see Chapter 3, "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

These are the supported name/value pairs:

```
ttStats [-h | -help]
ttStats [-V | -version]
ttStats [-monitor] [-interval seconds]
        [-duration seconds] [-iterations count]
          {DSN | -connStr connection_string}
ttStats  -snapshot [-level capture_level] [-description snap_desc]
          {DSN | -connStr connection_string}
ttStats  -report [-snap1 snapid1 -snap2 snapid2]
        [-html | -text] [-outputFile filename]
          {DSN | -connStr connection_string}
ttStats  -drop [-begin_snap snapid1 [-end_snap snapid2]]
          {DSN | -connStr connection_string}
```

> **Note:** Specify only one of -monitor, -snapshot, or -report.

Options

These are the supported options for the ttStats utility in TimesTen Scaleout:

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 -version	Prints the TimesTen release number and exits.
-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.

Option	Description
-description *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.
-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.
-drop	Delete snapshots in the range specified by -begin_snap and -end_snap (inclusive) from the system. This option is useful to keep the snapshot storage under the limit of 255 snapshots in the database.
-begin_snap *snapid1*	For the -drop option, this specifies the snapshot ID at the beginning of the range of snapshots to delete.
-end_snap *snapid2*	For the -drop option, this specifies the snapshot ID at the end of the range of snapshots to delete. If -end_snap is not specified, then only the snapshot specified by -begin_snap is deleted.
-connStr *connection_string* 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 database1

Connected to TimesTen Version 18.1.4.1.0 TimesTen Cache version 18.1.4.1.0.

Waiting for 10 seconds for the next snapshot
Description                     Current  Rate/Sec  Notes
date.2016-Feb-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                database1.ds0
loghold.checkpoint_hold_lsn      0/12023808                database1.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=database1;PLSQL_MEMORY_ADDRESS=20000000"
```

Snapshot example

The following examples take two snapshots at the default typical level:

```
% ttStats -snapshot database1

Connected to TimesTen Version 18.01.01.0001 TimesTen Cache version
18.1.4.1.0.
Snapshot 1 at TYPICAL level was successfully captured.

% ttStats -snapshot database1

Connected to TimesTen Version 18.1.4.1.0 TimesTen Cache version
18.1.4.1.0
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 database1

Connected to TimesTen Version 18.01.01.0001 TimesTen Cache version
18.1.4.1.0.
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 5–1 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 5–1   ttStats report: summary

Statement statistics   Figure 5–2 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 5–2   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 5–3 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 5–3   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 5–4 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 5–4   ttStats report: SQL execution statistics

SQL statistics: sort by preparations   Figure 5–5 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 5–5   ttStats report: SQL preparation statistics

SQL statistics: command texts   Figure 5–6 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 5–6   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 5–7 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 5–7   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 5–8 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 5–8   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 5–9 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 5–9   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 5–10 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 "Configure log buffer and log file size parameters" in *Oracle TimesTen In-Memory Database Operations Guide*.)

Figure 5–10   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 5–11 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 5–11   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 5–12 shows checkpoint metrics from a report.

Figure 5–12   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 5–13 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 5–13   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 5–14 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 5–14   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 5–15 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 5–15   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 5–16 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 5–16   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 5–17 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 5–17   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 5–18 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 2, "Connection Attributes".

Figure 5–18   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

See also

ttStatsConfig
ttStatsConfigGet

TimesTen Scaleout

This section describes how the `ttStats` utility works in TimesTen Scaleout.

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.

The `ttStats` utility 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, latches, locks, 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 and Linux systems 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: In TimesTen Scaleout, the `ttStats` daemon automatically takes snapshots of the TimesTen Scaleout database based on the parameters of the `ttStatsConfig` built-in procedure.

  If you use the `-snapshot` option of the `ttStats` utility, the `-description` option is mandatory. When you execute `ttStats -snapshot -description` *description*, you can associate a description to the latest system generated snapshot. Provide any description or notes for the snapshot, for example to distinguish it from other snapshots.

- Report mode: Generates a report from two specified snapshots or two specified timestamps of metrics. Reports are only available in plain text format. You must specify an output file with the `-outputFile` option. 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:
>
> - The utility cannot be used if you are connecting to TimesTen through a driver manager.

For information about built-in procedures mentioned, and the data they gather, see Chapter 3, "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.

- **Snapshot and report mode**: By default, only the instance administrator has privilege to create snapshots, get snapshot information and run in 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 connection_string}
ttStats  -snapshot -description snap_desc
            {DSN | -connStr connection_string}
ttStats  -report [-snap1 snapid1 -snap2 snapid2 |
        -timestamp1 'timestamp1' -timestamp2 'timestamp2']
        -outputFile filename
ttStats  -snapshotInfo {DSN | -connStr connection_string}
```

> **Note:** Specify only one of `-monitor`, `-snapshot`, or `-report`.

Options

These are the supported options for the `ttStats` utility in TimesTen Scaleout:

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 -version	Prints the TimesTen release number and exits.

Option	Description
-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 -description *snap_desc*	If you use the -snapshot option, the -description option is mandatory. When you execute ttStats -snapshot -description *description*, you can associate a description to the latest system generated snapshot. Provide any description or notes for the snapshot, for example to distinguish it from other snapshots.
-report	Generate a report from two specified snapshots or two specified timestamps. Use the -snapshotInfo option to see available snapshots for your database.
-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. The report period must span at least four existing snapshot ID values. Therefore, you must have at least three snapshots between -snap1 and -snap2.
-timestamp1 '*timestamp1*'	For report mode, this specifies the timestamp of the first snapshot. The timestamp must use the YYYY-MM-DD HH:MM:SS format and be wrapped in straight single quotes.
-timestamp2 'timestamp2'	For report mode, this specifies the timestamp of the second snapshot. The timestamp must use the YYYY-MM-DD HH:MM:SS format and be wrapped in straight single quotes.
-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.
-snapshotInfo	Prints the snapshot ID, date, time, and the description of all snapshots.
-connStr *connection_string* 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 database1

Connected to TimesTen Version 18.01.01.0001 Oracle TimesTen IMDB version
18.1.4.1.0.

Waiting for 10 seconds for the next snapshot

Description                    Current  Rate/Sec  Notes
date.2016-Mar-16 15:29:23    1458167363        1  sample #, not rate
connections.count                    20
lock.locks_granted.immediate     124817        1
log.log_bytes_per_transaction         0
loghold.bookmark.log_force_lsn 0/21102592
loghold.bookmark.log_write_lsn 0/21102856
loghold.checkpoint_hold_lsn    0/21100544           database1.ds0
loghold.checkpoint_hold_lsn    0/21078016           database1.ds1
plsql.GetHitRatio                 0.640    0.000
plsql.GetHits                   258.000    0.200
plsql.Gets                      403.000    0.200
plsql.PinHitRatio                 0.557    0.000
plsql.PinHits                   424.000    0.200
plsql.Pins                      761.000    0.200
stmt.executes.count              24407        1
stmt.executes.selects              620        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=database1"
```

Report examples

The following example creates a report from the snapshots generated in the previous section.

> **Note:** The report period must span at least four existing snapshot ID values. Therefore, you must have at least three snapshots between `-snap1` and `-snap2`.

```
% ttStats -report -outputFile testreport.txt -snap1 1 -snap2 5 database1

Connected to TimesTen Version 18.01.01.0001 Oracle TimesTen IMDB version
```

```
18.1.4.1.0.
Report testreport.txt was created.
```

The rest of this section shows excerpts from tables of metrics for a `ttStats` report for a TimesTen Scaleout database and a `ttStats` report for an element of a TimesTen Scaleout database. This output was produced using the default plain text output format.

- TimesTen Scaleout ttStats report
- TimesTen Scaleout element ttStats report

> **Note:** To include latch metrics, you must enable them for the database, using the `ttXactAdmin` utility as follows:
>
> `% ttXactAdmin -latchstats on DSN`

TimesTen Scaleout ttStats report

The following sections show excerpts from tables of metrics for a `ttStats` report for a TimesTen Scaleout database.

TimesTen Scaleout snapshot summary   Displays a grid snapshot summary. The TimesTen Scaleout snapshot summary shows information regarding the snapshots that you specified for the `ttStats` report.

TimesTen Scaleout elements   Displays information about each database element. This section of the report shows the host name and current number of connections for each database element. See "TimesTen Scaleout element ttStats report" on page 5-161 for more information on the contents of an elements `ttStats` report.

TimesTen Scaleout summary   Displays a summary of critical TimesTen Scaleout statistics. The summary includes statistics of your TimesTen Scaleout about transaction rates, SQL statements, database connections, checkpoint rates, transaction log rates, and other critical statistics.

TimesTen Scaleout load profile   Displays various database metric rates. This gives you an idea of the workload, showing the rate of checkpoints, log buffer waits (delays when the log buffer fills and flushes to disk), inserts, updates, deletes, parses (such as for prepares), commits, and s. Consider whether there may be too many parses or too many durable commits (which are more expensive than non-durable commits).

Operating system metrics summary   Displays various operating system metrics for the TimesTen Scaleout database. These metrics show the used disk space, CPU, I/O rate, and RAM.

TimesTen Scaleout efficiency metrics   Displays various metrics that determine the efficiency of the TimesTen Scaleout. This section of the report includes the following sections:

- Target 100% - bigger is better: This shows you recommendations to improve the efficiency of the TimesTen Scaleout. It includes the following metrics:
  - Prepare exec efficiency: This shows if your SQL statements are prepared and then executed many times. If you prepare a SQL statement once for every execution, this metric goes down.

Try to minimize the number of times your SQL statements are prepared because preparing SQL statements is CPU intensive. In your applications, consider using bind variables. You can then prepare your SQL statements once and then execute your SQL statements multiple times.

- Target 0% - smaller is better: This shows you recommendations to improve the efficiency of the TimesTen Scaleout. It includes the following metrics:

  – Log buffer waits: This shows the number of log buffer waits which helps you determine how operations that use the log files are doing. It is optimal to maintain the log buffer wait low because it indicates that transactions do not need to wait before writing to the log buffer.

  If this number is large, try to checkpoint more frequently, increase the Log Buffer Size and/or increase the log buffer parallelism

  – Table data skew deviation: This shows the percentage of table data skew deviations between the elements of the TimesTen Scaleout. Ideally the rows in tables are distributed evenly across all elements. If elements have too many rows compared to other elements, the elements with more rows use more `permSize` which can cause disk size and data distribution problems.

  – Direct mode connection distribution deviation: This shows the percentage of the direct mode connection deviation between the elements of the TimesTen Scaleout. Evenly spread the direct mode connections between the elements to achieve optimal throughput and latency.

  – Client server connection distribution deviation: This shows the percentage of the client/server connection deviation between the elements of the TimesTen Scaleout. Evenly spread the client/server connections between the elements to achieve optimal throughput and latency.

  – SQL statement distribution deviation: This shows the percentage of the SQL statement distribution deviation between the elements of the TimesTen Scaleout. It is not optimal to execute all SQL statements on a single attempt. Evenly execute the SQL statements on the elements to achieve optimal throughput and latency.

  – Grid channel invalidation: This shows the number of channel invalidations between the elements of the TimesTen Scaleout. Applications should cleanly disconnect and release resources to minimize channel invalidations. The cleanup process that TimesTen Scaleout performs after a channel invalidation takes time, which affects the latency and scalability of operations that want to use that channel.

TimesTen Scaleout transactions   Displays various metrics that show information about transactions on the TimesTen Scaleout. This section of the report includes the following sections:

- TimesTen Scaleout transactions per second

  This table shows various transaction metrics for each element such as the rate of transactions that: only involve the local element, multiple elements, and require 2PC.

- TimesTen Scaleout 2PC transactions

  This table shows various 2PC transaction metrics for each element such as the percentage of transactions that: started on this element, were involved in a 2PC transaction but did not initiate it, and used durable 2PC prepares.

- TimesTen Scaleout Durable Commits

This table shows various transaction metrics for each element such as the percentage of transactions that are committed durably on this element.

SQL statements: SQL statement protocol   Displays an excerpt of SQL statement protocol statistics from the SQL Statements section of a report. These statistics show you the percentage of SQL statements: executed for that element, executed on their local element, that required execution on a remote element, and that required a broadcast to all elements to execute.

SQL statements: SQL statements type   Displays an excerpt of SQL statement type statistics from the SQL Statements section of a report. These statistics show you various statistics of SQL statements executed for that element.

DB connections   Displays various connection statistics for each element of the TimesTen Scaleout. These statistics show you the type of connections, connections and disconnections per minute, and client server failover for every element of the TimesTen Scaleout.

TimesTen Scaleout data distribution: table data skew - worst three tables   Displays the three tables with the highest data skew percentage of the TimesTen Scaleout. For more information on the row distribution table, see "TimesTen Scaleout data distribution: row distribution for table" on page 5-160. These statistics show you the percentage of deviation, the table distribution type, and the distribution keys for the three tables with the highest data skew percentage.

TimesTen Scaleout data distribution: TimesTen Scaleout PermSize usage   Displays statistics related to the `PermSize` attribute for each element of the TimesTen Scaleout. These statistics show you the proportional percentage of used `PermSize` for each element in the TimesTen Scaleout, percentage of used `PermSize`, the percentage of used high water of `PermSize`, and the size of the `PermSize` for each element of the TimesTen Scaleout. For more information about the `PermSize` attribute, see "PermSize" on page 2-47.

TimesTen Scaleout data distribution: TimesTen Scaleout TempSize usage   Displays statistics related to the `TempSize` attribute for each element of the TimesTen Scaleout. These statistics show you the percentage of used `TempSize`, the percentage of used high water of `TempSize`, and the size of the `TempSize` for each element of the TimesTen Scaleout. For more information about the `TempSize` attribute, see "TempSize" on page 2-49.

TimesTen Scaleout data distribution: row distribution for table   There are three of these tables in your `ttStats` report, which show row distribution statistics for the tables with the highest data skew percentage of the TimesTen Scaleout. These statistics show you the number of rows that are stored on each element for that specific table. For more information about the three tables with the highest data skew percentage of the TimesTen Scaleout, see "TimesTen Scaleout data distribution: table data skew - worst three tables" on page 5-160.

TimesTen Scaleout channel: TimesTen Scaleout messages per second   Displays statistics related to message rates over TimesTen Scaleout channels. These messages can be requests for data or data result sets. These statistics show you the number of sent, received, and invalidated messages for each element of the TimesTen Scaleout.

TimesTen Scaleout channel: TimesTen Scaleout channel data rate   Displays statistics related to channel data rates for each element of the TimesTen Scaleout. These

data rates are based on the size of messages that TimesTen Scaleout sends over the channels. The messages can be requests for data or data result sets. Larger messages tend to have better throughput than smaller messages. These statistics show you the data rates of sent and received messages for each element of the TimesTen Scaleout.

Checkpoint: checkpoint data rate   Displays statistics related to checkpoint data rates for each element of the TimesTen Scaleout database.

Transaction Log: transaction log data rate   Displays statistics related to transaction log data rates for each element of the TimesTen Scaleout database.

Top SQL: top SQL attributes   Displays statistics related to the attributes of the most executed SQL statements on the TimesTen Scaleout database.

Top SQL: top SQL text   Displays information related to the SQL text of the most executed SQL statements on the TimesTen Scaleout database.

OS Disk Space   Displays the statistics related to the disk space of every element that is part of the grid.

CPU Utilization   Displays the statistics related to the CPU of every element that is part of the grid.

Operating System IO Rates: disk IO rates   Displays the statistics related to the disk I/O rates of every element that is part of the grid. This table enables you to determine if a host has a lot of disk I/O operations that are unrelated to TimesTen Scaleout.

Operating System IO Rates: network IO rates   Displays the statistics related to the network I/O rates of every element that is part of the grid. This table enables you to determine if a host has a lot of network I/O operations that are unrelated to TimesTen Scaleout.

OS Memory   Displays the statistics related to the memory usage of every element that is part of the grid. This table enables you to determine if a host is using swap space or is having performance issues due to a lack of available RAM.

TimesTen Scaleout element ttStats report

The following sections show excerpts from tables of metrics for a `ttStats` report for an element of a TimesTen Scaleout database.

Snapshot Summary   Displays the statistics related to the snapshots that `ttStats` uses to create the `ttStats` report for the element.

Element Summary   Displays a summary of critical statistics for the element. The summary includes statistics of your element related to transaction rates, SQL statements, database connections, checkpoint rates, transaction log rates, and other critical statistics.

Load Profile   Displays various database metric rates. This gives you an idea of the workload, showing the rate of checkpoints, log buffer waits (delays when the log buffer fills and flushes to disk), inserts, updates, deletes, parses (such as for prepares), commits, and s. Consider whether there may be too many parses or too many durable commits (which are more expensive than non-durable commits).

Operating System Metrics Summary   Displays various operating system metrics for the element. These metrics show the used disk space, CPU, I/O rate, and RAM.

Efficiency Metrics   Displays various metrics that determine the efficiency of the element. This section of the report includes the following sections:

- Target 100% - bigger is better: This shows you recommendations to improve the efficiency of the element. It includes the following metrics:

  - Prepare exec efficiency: This shows if your SQL statements are prepared and then executed many times. If you prepare a SQL statement once for every execution, this metric goes down.

    Try to minimize the number of times your SQL statements are prepared because preparing SQL statements is CPU intensive. In your applications, consider using bind variables. You can then prepare your SQL statements once and then execute your SQL statements multiple times.

- Target 0% - smaller is better: This shows you recommendations to improve the efficiency of the element. It includes the following metrics:

  - Log buffer waits: This shows the percentage of log buffer waits which helps you determine how operations that use the log files are doing. It is optimal to maintain the log buffer wait low because it indicates that transactions do not need to wait before writing to the log buffer.

    If this percentage is high, try to checkpoint more frequently, increase the Log Buffer Size and/or increase the log buffer parallelism

  - Table data skew deviation: This shows the percentage of table data skew deviations between the elements of the TimesTen Scaleout. Ideally the rows in tables are distributed evenly across all elements. If elements have too many rows compared to other elements, the elements with more rows use more `permSize` which can cause disk size and data distribution problems.

  - Direct mode connection distribution deviation: This shows the percentage of the direct mode connection deviation between the elements of the TimesTen Scaleout. Evenly spread the direct mode connections between the elements to achieve optimal throughput and latency.

  - Client server connection distribution deviation: This shows the percentage of the client/server connection deviation between the elements of the TimesTen Scaleout. Evenly spread the client/server connections between the elements to achieve optimal throughput and latency.

  - SQL statement distribution deviation: This shows the percentage of the SQL statement distribution deviation between the elements of the TimesTen Scaleout. It is not optimal to execute all SQL statements on a single attempt. Evenly execute the SQL statements on the elements to achieve optimal throughput and latency.

  - Grid channel invalidation: This shows the percentage of channel invalidations between the elements of the TimesTen Scaleout. Applications should cleanly disconnect and release resources to minimize channel invalidations. The cleanup process that the TimesTen Scaleout performs after a channel invalidation takes time, which affects the latency and scalability of operations that want to use that channel.

Transactions   Displays various metrics that show information about transactions on the element. This section of the report includes the following sections:

■ Transaction Type

This table shows various transaction metrics for your element such as the percentage of transactions that: only involve the local element, use remote transactions, and require 2PC.

■ 2PC transactions

This table shows various 2PC transaction metrics for each element such as the percentage of transactions that: started on this element, were involved in a 2PC transaction but did not initiate it, and used durable 2PC prepares.

SQL statements: SQL statement protocol   Displays an excerpt of SQL statement protocol statistics for the element from the SQL Statements section of a report. These statistics show you the percentage of SQL statements: executed, executed locally, that required execution on a remote element, and that required a broadcast to all elements to execute.

SQL statements: SQL statement type   Displays an excerpt of SQL statement type statistics for the element from the SQL Statements section of a report. These statistics show you various statistics for SQL statements executed on your element.

Database connections   Displays various connection statistics for the element. These statistics show you the type of connections, connections and disconnections per minute, and client server failover for your element.

Table data skew   Displays the three tables with the highest data skew percentage of the element. For more information on the row distribution table, see "TimesTen Scaleout data distribution: row distribution for table" on page 5-160. These statistics show you the percentage of deviation, the table distribution type, and the distribution keys for the three tables with the highest data skew percentage.

Grid channel usage   Displays message statistics over grid channels. These messages can be requests for data or data result sets. These statistics show you the number of sent, received, and invalidated messages for the element.

Log holds   Displays log hold information from a report. It shows bookmark positions for checkpoint log holds for each checkpoint file. 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.)

Checkpoint usage   Displays checkpoint usage metrics from a report.

Transaction log usage   Displays transaction log usage statistics for the element. This provides information about the rate of I/O operations for the transaction log, log buffer waits, log file reads, and log reads for commits.

Top SQL: top SQL attributes   Displays statistics related to the attributes of the most executed SQL statements on the element.

Top SQL: top SQL text   Displays information related to the SQL text of the most executed SQL statements on the element.

See also

ttStatsConfig
ttStatsConfigGet

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 attributes" 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] [-gridbrief]
ttStatus [-r secs] [-[no]pretty] {DSN | -connStr connection_string}
```

Options

ttStatus has the options:

Option	Description
-h -help -?	Prints usage information and exits.
-V	-version	Prints the release number of ttStatus and exits.
-connStr *connection_string*	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
DSN	An ODBC data source name of the database for which to get status.

Option	Description
-v	Prints detailed information that is useful for TimesTen customer support.
-r *secs*	Enables ttStatus to continue running. Updates status report every *secs* seconds.
-[no]pretty	The -pretty option (default) is for "pretty" formatting, which uses the values of the ConnectionName connection attribute. The -nopretty option is to not use pretty formatting.
-gridbrief	Prints a summary of all databases for the instance ttStatus is run from. Without this option, information is also shown for all connections.

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_18110 DSN:

```
% ttstatus cachedb1_18110

TimesTen status report as of Thu May 02 19:45:43 2013

Daemon pid 5280 port 53392 instance tt1811
TimesTen server pid 3940 started on port 53393
------------------------------------------------------------------------
Open for user connections
Data store cachedb1_18110
There are 12 connections to the data store
Shared Memory KEY Global\cachedb1_18110.c|. . .HANDLE 0x254
PL/SQL Memory KEY Global\cachedb1_18110.c|. . . HANDLE 0x258 Address 0x5B8C0000
Type            PID     Context     Connection Name          ConnID
Process         5196    0x01066a58  cachedb1_18110                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
```

Show a summary of the (only) TimesTen Scaleout database on the current instance:

```
% ttstatus -gridbrief
TimesTen status report as of Thu Feb  7 11:57:32 2019

Daemon pid 12767 port 6624 instance grid1_mgmt
TimesTen server pid 12819 started on port 6625
Grid GUID: C19E1FE5-2E58-4C3F-A66A-81661763B65F
Grid Membership Service is zookeeper
Grid Membership:
```

```
6 members
  1!mysys2!6624!0EC4823A-775B-4FD3-A33A-76C24D79E2EA!MGMT!18.1.2.1.0!
  1!mysys1!6624!2D6892CA-AA0C-488D-AB0D-20D0AF994018!MGMT!18.1.2.1.0!
  1!mysys3!46342!873678B9-297E-43A9-A183-2DDAD31E4391!DATA!18.1.2.1.0!
  1!mysys1!46344!F89F57B6-D904-4CD8-A6A0-9764A02C31F2!DATA!18.1.2.1.0!
  1!mysys1!46338!799287BA-FB41-47BD-A990-0B1745D5AD90!DATA!18.1.2.1.0!
  1!mysys2!46340!213D68A4-00AE-4429-A652-D2BC45C20AE8!DATA!18.1.2.1.0!
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Data store /sw/ttinstances/grid1_mgmt/grid/admin/database/ttgridadmin.0
Daemon pid 12767 port 6624 instance grid1_mgmt
TimesTen server pid 12819 started on port 6625
Replication policy  : Manual
Replication agent is running.
Cache Agent policy  : Manual
PL/SQL enabled.
-------------------------------------------------------------------------
Accessible by group timesten
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

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 5-169 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 that specifies a database location, driver, and optionally other connection attribute settings.
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 database1
```

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 that specifies a database location, driver, and optionally other connection attribute settings.
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" database1
	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	Prints a usage message and exits.
-help	
-?	
-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*.
tracefiles *n*	Sets the maximum number of output files.

Command	Description
tracefilesize *n*[M|G]	Sets the file size limit for output files.
	If M is specified indicates the file size in the indicated number of megabytes
	If G is specified indicates the file size in the indicated number of gigabytes.

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 options:

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.
config_found	Lists yes or no, indicating whether the configuration file is found.
product	Lists the name of the product.
major1	The first part of the five-part release number (18 for release 18.1.2.1.0), indicating the last two digits of the year of the major release.
	A change in major1 indicates major infrastructure and functionality changes.

Attribute	Description
major2	The second part of the five-part release number (1 for release 18.1.2.1.0).
	A change in only major2 indicates a version with new functionality and possibly some infrastructure changes.
	Releases with the same major1.major2 are patch-compatible. Data can be unloaded from a database from one release and loaded into a database from the other without the migration process.
patchset	The third part of the five-part release number (2 for release 18.1.2.1.0).
	A change in only patchset indicates a release that contains bug fixes and possibly some feature enhancements.
patch	The fourth part of the five-part release number (1 for release 18.1.2.1.0).
	A change in only patch indicates a release with only critical bug fixes.
reserved	The fifth part of the five-part release number (0 for release 18.1.2.1.0). Reserved for future use.
major3	No longer needed, but maintained for backward compatibility. Same as patchset.
portpatch	No longer needed, but maintained for backward compatibility. Same as reserved.
version	All five parts of the release number, separated by periods (such as 18.1.2.1.0).
shortversion	The first two parts of the five-part release number, without periods (181 for release 18.1.2.1.0).
numversion	All five parts of the release number, without periods, in the format %02d%02d%04d%02d%02d, compatible with ODBC version number format (180100020100 for release 18.1.2.1.0).
bits	Lists 64 to indicate the 64-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: tt1811.
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.
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.

Attribute	Description
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.
grid	Indicates if the instance is configured for grid distribution.
group_name	The name of the instance group.

Output

Following is ttVersion output without the -m option:

```
TimesTen Release 18.1.2.1.0 (64 bit Linux/x86_64) (grid1_mgmt:6624)
2019-01-15T06:25:33Z
  Instance admin: ttuser1
  Instance home directory: /sw/ttinstances/grid1_mgmt
  Group owner: timesten
  Daemon home directory: /sw/ttinstances/grid1_mgmt/info
  PL/SQL enabled.
```

And with the -m option:

```
patched=yes
config_found=yes
product=TimesTen
major1=18
major2=1
patchset=2
patch=1
reserved=0
major3=2
portpatch=0
version=18.1.2.1.0
shortversion=181
numversion=180100020100
bits=64
os=Linux/x86_64
buildtstamp=1547533533
buildtime=2019-01-15T06:25:33Z
clientonly=no
instance=grid1_mgmt
effective_port=6624
orig_port=6624
instance_admin=ttuser1
effective_insthome=/sw/ttinstances/grid1_mgmt
orig_insthome=/sw/ttinstances/grid1_mgmt
effective_daemonhome=/sw/ttinstances/grid1_mgmt/info
orig_daemonhome=/sw/ttinstances/grid1_mgmt/info
plsql=1
grid=0
group_name=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 connection_string}

ttXactAdmin -latch [-interval seconds] [-count iterations]
{DSN | -connstr connection_string}

ttXactAdmin -latch [-interval seconds] [-count iterations]
{DSN | -connstr connection_string}

ttXactAdmin -connections [-pid pid] [-interval seconds]
[-count iterations] {DSN | -connstr connection_string}

ttXactAdmin -xactIdRollback xid {DSN | -connstr connection_string}

ttXactAdmin -XactIdCommit xid

ttXactAdmin {-HCommit xid | -HAbort xid | -HForget xid} {DSN | -connstr
connection_string}
```

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 connection_string	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.

Option	Description
-count *iterations*	Generate the report iterations times. If no -interval option is specified, an interval of 1 second is used.
DSN	Indicates the ODBC data source name of the database to be administered.
	This option requires ADMIN privileges.
-h -help -?	Prints a usage message and exits.
-HAbort *xid*	Available for TimesTen Scaleout. 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.
	This option is not supported in TimesTen Scaleout.
-HCommit *xid*	Available for TimesTen Scaleout. 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.
	This option is not supported in TimesTen Scaleout.
-HForget *xid*	Available for TimesTen Scaleout. 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.
	This option is not supported in TimesTen Scaleout.
-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.
-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.
-tbl [*owner.*]*tableName*	Displays lock information for the specified table.
	This option requires ADMIN privileges or ownership of the specified table.
-V | -version	Prints the release number of ttXactAdmin and exits.
-v *verbosity*	Specifies the verbosity level. One of:
	0 - Does not display the names of the tables for row locks. In this case, ttXactAdmin runs faster.
	1 (default) - Displays the names of the tables for row locks.

Option	Description
-xact *xid*	Displays information for the specified transaction, including its log hold LSN.
	In the output, the field "Last ID" is a set of two sequence numbers. If the sequence numbers did not change in an interval, then no log record was written by the transaction during that interval.
	This option requires ADMIN privileges or ownership of the specified transactions.
-xactIdCommit *xid*	Available for TimesTen Scaleout. Enables you to commit 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.
-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 rollback 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.

Column	Description
XactID	The unique identifier for the transaction used internally by TimesTen. For TimesTen Classic, the identifier has two parts. For TimesTen Scaleout, the identifier is prefixed by the Element ID.
	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).
State	Current state 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.
	When using TimesTen Scaleout, the 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.
	`Grid-Doubtful-Yes` - The grid transaction prepared and voted yes for commit on this element, and is now doubtful.
	`Grid-Doubtful-No` - The grid transaction prepared and voted no for commit on this element, and is now doubtful.
	`Grid-Err` - The grid transaction returned an error on this element.

Column	Description
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.
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.

Column	Description
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"
2018-03-20 13:02:54.760
/timesten/jsmith/demo/demodata
TimesTen Release 18.1.1.1.0
ElementID 1

Program File Name: _ttIsql
XactID          PID      Context          State        Loghold Last ID
1.1.195         115640   0x859570         Active       391.15355904 [1666839:6]

  Resource  ResourceID          Mode  SqlCmdID          Name
  Database  0x01312d0001312d00  IX    0
  Table     2367528             IXn   275642480         JSMITH.T
  Row       AAAVVUAAADXAQAANje  Xn    275642480         JSMITH.T

  Begin Time: 13:01:43.108

1 outstanding transaction 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 a 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 connection_string	An ODBC connection string that specifies a database location, driver, and optionally other connection attribute settings.
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.logn, dspath is /home/payroll/2011.
-h -help -?	Prints a usage message and exits.
-headers records	Prints one header for every records records. A value of 0 disables headers.

Option	Description
-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*.

6

System Limits

The following sections list all TimesTen system limits and defaults.

- System limits and defaults
- Limits on number of open files
- Path names

System limits and defaults

Specific operating system limits may take precedence over these values.

For more information, see "Operating system prerequisites" in *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide* or "Operating system prerequisites" in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Description	Value
Maximum number of subscriber databases in a replication scheme that is not an active standby pair.	128
Maximum number of propagators in a replication scheme. Each propagator can have the maximum number of subscribers.	128
Maximum number of subscriber databases in an active standby pair.	127
Minimum database size (bytes). Size includes both the permanent and temporary space required to perform operations on the database.	32 MB
Maximum length for a fixed-length column (bytes).	8,300
Maximum number of columns in a table.	1,000
Maximum number of columns in an `ORDER BY` clause.	1,000
Maximum number of columns in a `GROUP BY` clause.	1,000
Maximum cumulative length of a row's fixed-length columns (bytes).	32,768
Maximum length for a varying-length column (bytes).	$2^{22} = 4,194,304$
Maximum length for a replicated column.	4 MB 16 MB for columns with `BLOB` type
Maximum number of concurrent connections to a database (including system connections).	32,047

Description	Value
Maximum number of concurrent application connections to a database (may be limited by semaphore configuration or `Connections` DSN attribute or both).	32,000 2,000 (default)
Maximum number of connections (system and application) across all databases in an instance.	32,048
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
Maximum length of database names.	32
Maximum length of the path name for a database in an asynchronous writethrough cache group	248
Maximum number of projected expressions in a `SELECT` statement.	32,767
Maximum length of string specifying a join order.	1,024
Maximum number of columns in an index (or primary key).	32
Maximum length of basic names.	30
Maximum length of displayed predicate string in the `SYS.PLAN` table.	1,024
Maximum length of SQL statement, including the `NULL` terminator.	409,600
Maximum number of table references in an SQL query.	24
Maximum number of indexes on a table.	500
Maximum number of partitions in a table.	999
Maximum number of prepared PL/SQL statements per connection.	5000
Maximum number of recently-used PL/SQL blocks that can be cached per session.	5000
Maximum number of concurrent segment client/server connections per TimesTen instance	Unlimited (up to the limits of the operating system)
Maximum number of concurrent shared memory segment client/server connections per TimesTen instance.	Unlimited (up to the limits of the operating system)
Maximum size of IPC shared memory segment for client/server connections	4 GB
Maximum number of allocated statement handles per shared memory segment client/server connection.	512
Maximum depth of nesting subqueries.	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
Maximum number of replicated XLA bookmarks.	64

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 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