Oracle TimesTen In-Memory Database API Reference Guide

Release 7.0

B31683-01

For last-minute updates, see the TimesTen release notes.

February 2007

Printed in the United States of America

Contents

3 Built-In Procedures

About this Guide

Oracle TimesTen In-Memory Database is a high-performance, in-memory data manager that supports the ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity) interfaces.

This guide is for application developers who use and administer TimesTen and for system administrators who configure and manage the TimesTen. It provides a reference of all TimesTen utilities, procedures, APIs and other reference information.

To work with this guide, you should understand how database systems work. You should also have knowledge of SQL (Structured Query Language) and either ODBC or JDBC. See “Background reading” on page 5 if you are not familiar with these interfaces.

Conventions used in this guide

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

TimesTen documentation uses these typographical conventions:

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

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

If you see...	It means...
`fixed width italics`	Variable; must be replaced with an appropriate value.
`[ ]`	Square brackets indicate that an item in a command line is optional.
`{ }`	Curly braces indicated that you must choose one of the items separated by a vertical bar (|) in a command line.
`|`	A vertical bar (or pipe) separates arguments that you may use more than one argument on a single command line.
`...`	An ellipsis (. . .) after an argument indicates that you may use more than one argument on a single command line.
`%`	The percent sign indicates the UNIX shell prompt.
`#`	The number (or pound) sign indicates the UNIX root prompt.

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

If you see...	It means...
install_dir	The path that represents the directory where the current release of TimesTen is installed.
TTinstance	The instance name for your specific installation of TimesTen. Each installation of TimesTen must be identified at install time with a unique alphanumeric instance name. This name appears in the install path. The instance name "giraffe" is used in examples in this guide.
bits or *bb*	Two digits, either 32 or 64, that represent either the 32-bit or 64-bit operating system.
release or *rr*	Two digits that represent the first two digits of the current TimesTen release number, with or without a dot. For example, 70 or 7.0 represents TimesTen Release 7.0.
jdk_version	Two digits that represent the version number of the major JDK release. Specifically, 14 represent JDK 1.4; 5 represents JDK 5.
timesten	A sample name for the TimesTen instance administrator. You can use any legal user name as the TimesTen administrator. On Windows, the TimesTen instance administrator must be a member of the Administrators group. Each TimesTen instance can have a unique instance administrator name.
DSN	The data source name.

This guide uses these icons to identify platform-specific information:

If you see	It means
	Information applies to TimesTen on a supported UNIX platform.
	Information applies to TimesTen on the Windows 2000, Windows XP and Windows 2003 platforms.

	Information applies to TimesTen on the HP-UX platform.
	Information applies to TimesTen on the Solaris platform.
	Information applies to TimesTen on Tru64 platform.

Background reading

For a Java reference, see:

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

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

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

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

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

For a review of SQL, see:

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

For information about Unicode, see:

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

Technical Support

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

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

1

Data Store Attributes

The ODBC standard defines four data store 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 attributes defined by TimesTen. The table below gives a brief description and the default value for each attribute. The sections that follow provide detailed information about each attribute.

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

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

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.

To view the names and values of attributes specified in the connection string, an application can use the **ttConfiguration** built-in procedure.

Access Control and attributes

Only the instance administrator or a user with ADMIN privileges can change a first connection attribute to a value other than the one currently in effect.

In addition, Authenticate and Group Restrict require ADMIN privileges.

TimesTen determines privileges at connect time and they remain in effect until disconnect. Changes to or revocation of privileges for a user do not take effect until the user makes a new connection.

List of Attributes

Name	Description	Default
Data store attributes		
Authenticate (See page 14.)	Specifies that a Client connection to a Server DSN requires user and password authentication.	True
Data Source Name (See page 15.)	Identifies the attributes to a connection.	None
DataStore (See page 16.)	Identifies the physical data store.	None
DatabaseCharacterSet (See page 16.)	Identifies the character set used by the data store. This attribute is required at data store creation time.	None
Description (See page 17.)	A statement that identifies the use of the data store.	None
GroupRestrict (See page 17.)	Restricts data store access to members of the specified group.	None
LogDir (See page 18.)	Specifies the directory where log files are stored.	Data store directory
Preallocate (See page 19.)	Specifies that disk space for the data store should be preallocated when creating the data store.	False
Temporary (See page 20.)	Specifies that the data store is not saved to disk.	False
TypeMode (See page 20.)	Indicates the type mode for the data store.	**0** - Oracle Type Mode

First connection attributes		
Name	**Description**	**Default**
AutoCreate (See page 22.)	Specifies that the first connection creates the data store if it does not exist already.	True
CkptFrequency (See page 22.)	Controls the frequency in seconds that TimesTen performs a background checkpoint.	600 if **Logging**=1 is specified, otherwise it is 0 (off)

First connection attributes		
Name	**Description**	**Default**
CkptLogVolume (See page 24.)	Controls the amount of data in megabytes that collects in the log between background checkpoints.	0 (off)
CkptRate (See page 25.)	Controls the maximum rate at which data should be written to disk during a checkpoint operation.	0 (unlimited rate)
Connections (See page 27.)	Indicates the expected upper bound on the number of concurrent connections to the data store.	0 (64 connections.)
ForceConnect (See page 28.)	Specifies whether a connection is allowed to a failed data store if it is not properly restored from the corresponding subscriber data store.	0 (Connection disallowed)
LogAutoTruncate (See page 29.)	Determines whether the first connection to a data store should proceed if TimesTen recovery encounters a defective log record	1 (Continues after log is truncated)
LogBuffSize (See page 30.)	Specifies the size of the internal log buffer.	64 MB
LogFileSize (See page 31.)	Specifies the log file size in MB.	64 MB
LogFlushMethod (See page 32.)	Controls the method TimesTen uses to write and sync log files.	1 (Buffered writes)
Logging (See page 33.)	Specifies whether logging to disk or no logging should be performed for the data store.	1 (Logging to disk)
LogPurge (See page 34.)	Specifies that unneeded log files are deleted during a checkpoint operation.	True
MemoryLock (See page 35.)	Allows applications that connect to a shared data store to specify whether the real memory should be locked during data store loading	0 (Do not acquire a memory lock)
Overwrite (See page 36.)	Specifies that the existing data store should be overwritten with a new one when a connection is attempted.	False

First connection attributes		
Name	**Description**	**Default**
PermSize (See page 37.)	Specifies the size in MB for the permanent partition of the data store. For 32-bit systems: For 64-bit systems:	 2 MB 4 MB
RecoveryThreads (See page 38.)	Specifies the number of threads used to rebuild indexes during recovery.	1
TempSize (See page 39.)	Specifies the size in MB for the temporary partition of the data store.	The default size is determined from the PermSize value.

General connection attributes		
Name	**Description**	**Default**
ConnectionName (See page 40.)	Specifies whether there is a symbolic name for the data source.	The process name
Diagnostics (See page 41.)	Specifies whether diagnostic messages are generated.	1 (Messages are generated.)
DurableCommits (See page 42.)	Specifies that commit operations should write log records to disk.	0 (Records not written to disk)
Isolation (See page 43.)	Specifies whether the isolation level is read committed or serializable.	1 (Read committed)
LockLevel (See page 44.)	Specifies whether the connection should use row-level locking (value = 0) or data store-level locking (value = 1).	Row-level locking
LockWait (See page 45.)	Allows an application to configure the lock wait interval for the connection.	10 seconds
MatchLogOpts (See page 46.)	Specifies that values used for the **Logging** and **LogPurge** attributes should match those of current connections.	False

General connection attributes		
Name	**Description**	**Default**
PermWarnThreshold (See page 47.)	Specifies the threshold at which TimesTen returns a warning and throws an SNMP trap when the permanent partition of the data store is low in memory.	90%
PrivateCommands (See page page 48.)	Determines if commands are shared between connections.	0 (on)
PWD (See page 52.)	Required if the corresponding Server DSN is defined with **Authenticate** =1 or if Access Control is enabled and you attempt a connection as an internal user. Specify the password that corresponds with the specified UID. When caching Oracle data, PWD specifies the TimesTen password while OraclePWD specifies the Oracle password.	None
PWDCrypt (See page page 49.)	Specifies the value of the encrypted user password.	None
SqlQueryTimeout (See page 50.)	Specifies the query timeout value in seconds	0
TempWarnThreshold (See page 51.)	Specifies the threshold at which TimesTen returns a warning and throws an SNMP trap when the temporary partition of the data store is low in memory.	90%
UID (See page 52.)	Required if the corresponding Server DSN is defined with **Authenticate** =1. Specify a user name that is defined on the server machine. When caching Oracle data, the UID must match the UID on the Oracle database that is being cached in TimesTen.	None
WaitForConnect (See page 53.)	Specifies that the connection attempt should wait if an immediate connection is not possible.	True

NLS general connection attributes

Name	Description	Default
ConnectionCharacterSet (See page 54.)	Specifies the character encoding for the connection, which may be different from the database character set.	US7ASCII unless the database character set is TIMESTEN8, then TIMESTEN8.
NLS_LENGTH_SEMANTICS (See page 55.)	Sets the default length semantics configuration.	BYTE
NLS_NCHAR_CONV_EXCP (See page page 55.)	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.	False
NLS_SORT (See page 56.)	Indicates the collating sequence to use for linguistic comparisons.	BINARY

Cache Connect attributes

Name	Description	Default
OracleID (See page 62.)	Specifies the Service Identifier (SID) of the Oracle instance from which data is to be loaded into a TimesTen data store. The OracleID attribute is only used by the cache agent. Set the OracleID to the Oracle SID or the Oracle Service Name.	None
OraclePWD (See page 63.)	Password to the Oracle database that is being cached in TimesTen.	None
PassThrough (See page 64.)	Specifies which SQL statements are executed against the TimesTen data store and passed through to the Oracle database associated with a TimesTen cache group.	0

Cache Connect attributes

Name	Description	Default
RACCallback (See page 68.)	Specifies whether to enable or disable the installation of Transparent Application Failover (TAF) and Fast Application Notification (FAN) callbacks.	1 (Install the callbacks.)
TransparentLoad (See page 69.)	Specifies whether to enable or disable loading Oracle data automatically on SELECT when a query does not find data in a cache group.	0 (Disabled)

Client connection attributes

Name	Description	Default
TCP_Port (See page 58.)	Required if the TimesTen Server is not listening on the default port number and you are not using a logical server name for TTC_Server. If a logical server name is used, this attribute in the connection string takes precedence over the port number defined in the logical server name.	None
TTC_Server (See page 59.)	Required. Hostname or network address. Name of the machine where the TimesTen Server is running or a logical server name.	None
TTC_Server_DSN (See page 60.)	Required. Server DSN corresponding to the TimesTen data store.	None
TTC_Timeout (See page 61.)	Optional. Timeout period, in seconds, for completion of a TimesTen client/server operation. The maximum timeout period is 99999 seconds.	60 seconds

Data store attributes

Data store attributes are set at data store creation time.The data store attributes are listed in the table "Data store attributes" on page 8 and described in detail in this section.

Authenticate

Set this attribute in a Server DSN to require User ID and password authentication for all connections from the TimesTen Client driver to this DSN. The authentication is done by checking whether the User ID and password supplied by the TimesTen Client application/DSN can log onto the system where the TimesTen Server and TimesTen Data Manager are running.

This attribute is ignored when you use the TimesTen Data Manager to connect to a local DSN or if it is specified in the connection string of a connect call to a TimesTen Client DSN. In other words, you cannot overwrite the value of this attribute stored in the ODBC.INI file on UNIX or the ODBC Data Source Administrator on Windows.

To cache Oracle tables, Authenticate must be turned off. Cached tables must have the UID and OraclePWD attributes set to the Oracle login and password.

For information on working with Client and Server DSNs, see "Working with the TimesTen Client and Server" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect. To use TimesTen Client/Server when Access Control is enabled, Authenticate=1 must be set.

Setting

Set Authenticate as follows:

C or Java programs or UNIX ODBC.INI file	Authenticate	**0**—Does not authenticate User ID and password supplied by TimesTen Client application/DSN. **1**—Authenticates User ID and password supplied by TimesTen Client application/DSN (default).
Windows ODBC Data Source Administrator	**Authenticate** TimesTen Client Connection check box	**unchecked**—Does not authenticate User ID and password supplied by TimesTen Client application/DSN. **checked**—Authenticates User ID and password supplied by TimesTen Client application/DSN (default).

Data Source Name

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

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

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

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

Setting Set Data Source name as follows:

C or Java programs or UNIX ODBC.INI file	DSN	A name that describes the DSN.
Windows ODBC Data Source Administrator	**Data Source Name field**	A name that describes the DSN.

DataStore

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

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

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

Setting

Set DataStore as follows:

C or Java programs or UNIX ODBC.INI file	DataStore	Full path to the physical data store that the data source name references.
Windows ODBC Data Source Administrator	**Data Store Path** + **Name** field	Full path to the physical data store that the data source name references.

DatabaseCharacterSet

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

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

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

You should choose a connection character set (See “ConnectionCharacterSet” on page 54.) that matches your terminal settings or data source.

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

To use this attribute you must specify a supported character set. For a list of character set names that can be used as a value for this attribute, see "Supported Character Sets" in the *Oracle TimesTen In-Memory Database Operations Guide*. The Oracle TimesTen In-Memory Database supports the same list of character sets as Oracle XE, with the additional value TIMESTEN8 for the TimesTen legacy character set.

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

Setting Set DatabaseCharacterSet name as follows:

C or Java programs or UNIX ODBC.INI file	DatabaseCharacterSet	Specify the preferred character set.
Windows ODBC Data Source Administrator	**Database Character Set** list	Select the preferred character set from the list provided in the ODBC Data Source Administrator.

See also "ConnectionCharacterSet" on page 54.

Description

Optionally, set this attribute to help you identify the DSN and its attributes.

Setting Set Description as follows:

C or Java programs or UNIX ODBC.INI file	Description	Text description of the Data Source Name. This attribute is optional.
Windows ODBC Data Source Administrator	**Description field**	Text description of the Data Source Name. This attribute is optional.

GroupRestrict

The **GroupRestrict** attribute allows you to restrict data store access to a specific operating system user group. Specifying this attribute restricts all file and shared memory accessibility to a data store.

Note: TimesTen supports the creation of individual internal users, but not internal user groups.

Once set, the restrictions cannot be removed from a data store without destroying a data store.

If a data store is restricted, the TimesTen daemon must be run with root or administrative privileges. With a few exceptions, only group users can run a TimesTen utility against a group restricted data store.

With Client/Server data stores, if GroupRestrict is set, TimesTen sets Authenticate to 1.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect. The instance administrator must be a member of the operating system group specified by this attribute. On Windows, the instance administrator must be a member of the Administrators group

Setting

Set GroupRestrict as follows:

C or Java programs or UNIX ODBC.INI file	GroupRestrict	Specify the name of the operating system group of users who are allowed access to this data store
Windows ODBC Data Source Administrator	OS Group Restrict field	Specify the name of the operating system group of users who are allowed access to this data store

LogDir

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

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

If logging is off, TimesTen ignores the value of LogDir.

Setting Set LogDir as follows:

C or Java programs or UNIX ODBC.INI file	LogDir	Specifies the directory where log files reside.
Windows ODBC Data Source Administrator	Log Directory field	Specifies the directory where log files reside.

Preallocate

The **Preallocate** attribute determines whether TimesTen preallocates file system space for the data store when the data store is created. Setting this attribute ensures that there will be sufficient space for the data store when the data store is saved to the file system.

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

Access Control If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting Preallocate as follows:

C or Java programs or UNIX ODBC.INI file	Preallocate	**0**—Does not preallocate file system space for data store when creating the data store (default). **1**—Preallocates file system space for the data store.
Windows ODBC Data Source Administrator	**Preallocate** check box	**unchecked**—Does not preallocate file system space for data store when creating the data store (default). **checked**—Preallocates file system space for the data store.

Note Preallocating disk space for a large data store is very time consuming.

Temporary

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

Note: You cannot backup or replicate a temporary data store.

Setting Set Temporary as follows:

C or Java programs or UNIX ODBC.INI file	Temporary	**0**—Creates permanent data store (default). **1**—Creates temporary data store.
Windows ODBC Data Source Administrator	**Temporary** check box	**unchecked**—Creates permanent data store (default). **checked**—Creates temporary data store.

TypeMode

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

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

Setting Set TypeMode as follows:

C or Java programs or UNIX ODBC.INI file	TypeMode	**0** - Oracle Type Mode (default) **1** - TimesTen Type Mode If not specified, either the default type mode or the type mode assigned when the data store was created is used.
Windows ODBC Data Source Administrator	**TypeMode** dropdown list	**0** - Oracle Type Mode (default) **1** - TimesTen Type Mode If left blank, either the default type mode or the type mode assigned when the data store was created is used.

First connection attributes

First connection attributes are set when a connection is made to an idle data store (a data store with no connections) and persist for that connection and all subsequent connections until the last connection to this data store is closed.

First connection attributes are listed in the table "First connection attributes" on page 8 and described in detail in this section.

If you try to connect to the data store using attributes that are different from the first connection attribute settings, the new connection may be rejected or the attribute value may be ignored. For example, if existing connections have logging to disk enabled, a new connection cannot have logging to disk disabled. 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 or a user with ADMIN privileges can change a first connection attribute to a value other than the one currently in effect.

AutoCreate

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

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set AutoCreate as follows:

C or Java programs or UNIX ODBC.INI file	AutoCreate	**0**—Does not create new data store if data store does not exist. **1**—Creates new data store if data store does not exist (default).
Windows ODBC Data Source Administrator	**AutoCreate** check box	**unchecked**—Does not create new data store if data store does not exist. **checked**—Creates new data store if data store does not exist (default).

CkptFrequency

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

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

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

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

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting Set CkptFrequency as follows:

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

CkptLogVolume

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

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

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

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

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set CkptLogVolume as follows:

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

CkptRate

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

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

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

The value of this attribute is "sticky" as it persists across data store loads and unloads unless it is explicitly changed. The default value is only used during data store creation. Subsequent first connections default to using the existing value stored in the data store. If left unspecified (or empty in the Windows ODBC Data Source Administrator), the stored setting is used. To turn the attribute off, you must explicitly specify a value of 0. For existing data stores 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 for background checkpoints," in the *Oracle TimesTen In-Memory Database Operations Guide*.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set CkptRate as follows:

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

Connections

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

The number of current connections to a data store can be determined by viewing the output from the **ttStatus** utility.

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

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set Connections as follows:

C or Java programs or UNIX ODBC.INI file	Connections	**0**—Assumes 64 maximum connections (default). **An integer from 1 through 2047** — The value represents the expected maximum number of connections.
Windows ODBC Data Source Administrator	**Connections** field	**0**—Assumes 64 maximum connections (default). **An integer from 1 through 2047** — The value represents the expected maximum number of connections.

ForceConnect

When return receipt replication is used with the NONDURABLE TRANSMIT option, a failed master data store is allowed to recover only by restoring its state from a subscriber data store using the `-duplicate` option of the ttRepAdmin utility. In other words, the failed data store 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.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set ForceConnect as follows:

C or Java programs or UNIX ODBC.INI file	ForceConnect	**0**—Do not allow connection to failed data store if it is not properly restored from the corresponding subscriber data store (default). **1**—Allow connection to a failed data store even if it is not properly restored from the corresponding subscriber data store.
Windows ODBC Data Source Administrator	**ForceConnect** check box	**unchecked**—Do not allow connection to failed data store if it is not properly restored from the corresponding subscriber data store (default). **checked**—Allow connection to a failed data store even if it is not properly restored from the corresponding subscriber data store.

LogAutoTruncate

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

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set LogAutoTruncate as follows:

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

LogBuffSize

The LogBuffSize attribute specifies the size of the internal log buffer in kilobytes. The default log buffer size is 65,536 KB and the minimum is 128 KB. If you enter a size smaller than 128 KB, TimesTen overrides the value and creates a 128 KB internal log buffer.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set LogBuffSize as follows:

C or Java programs or UNIX ODBC.INI file	LogBuffSize	**n**—Size of log buffer, in kilobytes. If not set, TimesTen uses a default of 65,536 KB (64 MB).
Windows ODBC Data Source Administrator	**Log Buffer Size (KB)** field	Size of log buffer, in kilobytes. If you leave this field blank, TimesTen uses a default f 65,536 KB (64 MB).

LogFileSize

The **LogFileSize** attribute specifies the maximum size of log files. The default value is 64 MB. This attribute is relevant only if logging to disk is enabled. Actual log file sizes may be slightly smaller or larger than LogFileSize because log records are not allowed to span log files.

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

It is best to set the value of LogFileSize to match or exceed the **LogBuffSize**, even thought it is possible that the LogBuffSize value can be greater than the value of LogFileSize. Be careful though, because LogFileSize is sized in MB while LogBuffSize is sized in KB. There cannot be more then LogFileSize outstanding data in the log buffer so if the buffer is not sized to match the LogFileSize, the buffer may not get fully utilized.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set LogFileSize as follows:

C or Java programs or UNIX ODBC.INI file	LogFileSize	**n**—Size of log file, in megabytes: Default is 64 MB when the data store is created and 0 (current size in effect) on subsequent connections. The maximum size is 1G.
Windows ODBC Data Source Administrator	**Log Files Size (MB)** field	If you leave this field blank, a default of 64 MB is used when the data store is created and 0 (the current file size in effect) on subsequent connections. The maximum size is 1G.

LogFlushMethod

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

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

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

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set LogFlushMethod as follows:

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

Logging

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

If Logging is disabled, durable commits are not possible. If Logging is turned off, operations that are not atomic return an error or warning when the TimesTen Data Manager cannot restore the data store to its state prior to a failed operation. For more details on transaction management, see Chapter 7, "Transaction Management and Recovery," in the *Oracle TimesTen In-Memory Database Operations Guide*.

Some applications that can trade off durability and atomicity for performance may want to turn off logging entirely.This is something that should be carefully considered, as disabling logging restricts your application from using row-level locking, rolling back transactions, using Cache Connect to Oracle or Replication. Disabling logging may also be useful for improving performance of a bulk load operation if the operation can be restarted from the beginning in case of a failure.

To turn off logging entirely, set the **Logging** parameter to 0. To cache Oracle tables, you must enable logging to disk.

Row-level locking is disabled if **Logging** is turned off (**Logging** is set to 0). See "LockLevel" on page 44 for additional information. In addition, **DurableCommits** and **LogPurge**) are disabled when the **Logging** attribute is set to 0. Use the **MatchLogOpts** attribute to prevent such conflicts.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set Logging as follows:

C or Java programs or UNIX ODBC.INI file	Logging	**0**—Does not log modifications to the data store. **1**—Logs modifications to the data store to disk (default).
Windows ODBC Data Source Administrator	**Logging** field	**0**—Does not log modifications to the data store. **1**—Logs modifications to the data store to disk (default).

LogPurge

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

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

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

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

This attribute is relevant only if **Logging** is set to 1. See also "MatchLogOpts" on page 46.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set LogPurge as follows:

C or Java programs or UNIX ODBC.INI file	LogPurge	**0**—Does not remove old log files at connect and checkpoint. **1**—Removes old log files at connect and checkpoint (default).
Windows ODBC Data Source Administrator	**LogPurge** check box	**unchecked**—Does not remove old log files at connect and checkpoint. **checked**—Removes old log files at connect and checkpoint (default).

MemoryLock

On Solaris, Linux, HP-UX 11 and Tru64 systems, TimesTen allows applications that connect to a shared data store to specify whether the real memory should be locked while the data store is being loaded into memory or while the store is in memory. If the physical memory used for the data store is locked, the operating system's virtual memory sub-system cannot borrow that memory for other uses. No part of the data store will ever be paged out but this could lead to memory shortages in a system that is under configured with RAM. While memory locking can improve data store load performance, it may impede other applications on the same machine.

Access Control and Non-root instances

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

On Linux and Tru64 systems, root privileges are required to use this attribute. On Solaris systems, you must be installed as root to use MemoryLock=**1** or **2**.Data stores in a non-root instance of TimesTen can use settings **3** and **4** for this attribute, on Solaris systems.

Setting Set MemoryLock as follows:

C or Java programs or UNIX ODBC.INI file	MemoryLock	**0**—Does not lock memory (default). **1**— Tries to obtain a memory lock. If unable to lock, the connection succeeds. If a lock is obtained, it is released after the data store 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 data store 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 data store is unloaded from memory. **4**—A memory lock is required and is held until the data store 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 data store is unloaded from memory.
Windows ODBC Data Source Administrator		Not available.

Overwrite

If the **Overwrite** attribute is set and there is an existing data store with the same data store path name as the new data store, TimesTen destroys the existing data store and creates a new empty data store, as long as the existing data store is not in use. If the **Overwrite** attribute is set and there is not a data store with the specified data store path name, TimesTen only creates a new data store if the **AutoCreate** attribute is also set (see "AutoCreate" on page 22). Overwrite is ignored if AutoCreate is set to 0. Applications should use caution when specifying the **Overwrite=1** attribute.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set Overwrite as follows:

C or Java programs or UNIX ODBC.INI file	Overwrite	**0**—Does not overwrite existing data store with the same path name (default). **1**—Overwrites existing data store with the same path name.
Windows ODBC Data Source Administrator	**Overwrite** check box	**unchecked**—Does not overwrite existing data store with the same path name (default). **checked**—Overwrites existing data store with the same path name.

PermSize

Specifically indicates the size in MB of the permanent memory region for the data store. 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 data store does not exist, a PermSize value of 0 or no value indicates that the default size should be used. For an existing data store, a value of 0 or no value indicates that the existing size should not be changed.

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

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

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set PermSize as follows:

C or Java programs or UNIX ODBC.INI file	PermSize	**n**—Size of permanent partition of the data store, in megabytes; default is 2 MB for 32-bit systems and 4 MB for 64-bit systems. Minimum size is 2 MB.
Windows ODBC Data Source Administrator	**Permanent Data Size** field	**n**—Size of permanent partition of the data store, in megabytes; default is 2 MB for 32-bit systems and 4 MB for 64-bit systems. Minimum size is 2 MB.

RecoveryThreads

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

If RecoveryThreads=1, during recovery, indexes that need to be rebuilt are done serially. If you have enough processors available to work on index rebuilds on your machine, setting this attribute 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.

Access Control

If Access Control is enabled, only the instance administrator or a user with ADMIN privileges can change this attribute to a value other than the one currently in effect.

Setting

Set RecoveryThreads as follows:

C or Java programs or UNIX ODBC.INI file	RecoveryThreads	**n**—The number of threads to use to rebuild indexes during recovery. Default is 1.
Windows ODBC Data Source Administrator	**RecoveryThreads** field	**n**—The number of threads to use to rebuild indexes during recovery. Default is 1.

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 server machine for other services/processes.

TempSize

TempSize indicates the total amount of memory in MB allocated to the temporary region. TempSize has no pre-defined fixed default value. If left unspecified, its value is determined from PermSize as follows:

If PermSize is less than 64MB, TempSize = 6 MB + ceiling(PermSize / 4 MB).

If PermSize is greater than 64MB, TempSize =14 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 data store is loaded, the TempSize attribute may be increased or decreased each time a data store is loaded. For an existing data store, a value of 0 or no value indicates that the existing size should not be changed.

Setting Set TempSize as follows:

C or Java programs or UNIX ODBC.INI file	TempSize	**n**—Size of temporary partition of the data store, in megabytes; The default size is determined from the PermSize value. Minimum size is 6 MB.
Windows ODBC Data Source Administrator	**Temporary Data Size** field	**n**—Size of temporary partition of the data store, in megabytes; The default size is determined from the PermSize value. Minimum size is 6 MB.

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 the table "General connection attributes" on page 10 and described in detail in this section.

ConnectionName

This attribute is also available as a Client connection attribute.

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

The symbolic name is used to help identify the connection in various TimesTen administrative utilities, such as **ttIsql**, **ttXactAdmin** and **ttStatus**. This can be particularly useful with processes that make multiple connections to the data store, as is typical with multi-threaded 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.

Setting Set ConnectionName as follows:

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

Diagnostics

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

Setting Set Diagnostics as follows:

C or Java programs or UNIX ODBC.INI file	Diagnostics	**0**—No diagnostics messages are generated. **1**—Base level diagnostics messages are generated. (default).
Windows ODBC Data Source Administrator	**Diagnostics field**	**0**—No diagnostics messages are generated. **1**—Base level diagnostics messages are generated. (default).

DurableCommits

By default, **DurableCommits** is set to 0, meaning the log is not written to disk at transaction commit. This reduces the transaction execution time at the risk of losing some committed transactions in the event of a failure. However, if you are logging to disk, the log is saved to disk when the log buffer in memory fills up, regardless of the setting of **DurableCommits**.

A connection that is logging to disk 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 managing subdaemon processes also frequently flush the log buffer to disk. The log buffer is shared among all connections and contains log records from transactions of all connections.

If logging to disk is not enabled, durable commits are not possible. If logging to disk is disabled, durability of data can be achieved through checkpointing, which happens in the background at intervals determined by the settings of **CkptFrequency** and **CkptLogVolume** or which the application can explicitly initiate. TimesTen automatically initiates checkpointing when the last connection to the data store exits successfully.

Setting Set DurableCommits as follows:

C or Java programs or UNIX ODBC.INI file	DurableCommits	**0**—Does not force log to disk on transaction commit (default). **1**—Forces log to disk on transaction commit.
Windows ODBC Data Source Administrator	**Durable Commits** check box	**unchecked**—Does not force log to disk on transaction commit. **checked**—Forces log to disk on transaction commit

Isolation

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

If the passthrough or the propagate Cache Connect feature is used, the TimesTen's 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.

Setting Set Isolation as follows:

C or Java programs or UNIX ODBC.INI file	Isolation	**0**—Connects to data store in serializable isolation mode. **1**—Connects to data store in read-committed mode (default).
Windows ODBC Data Source Administrator	**Isolation** dropdown list	**0**—Connects to data store in serializable isolation mode. **1**—Connects to data store in read-committed isolation mode (default).

LockLevel

By default, TimesTen enables row-level locking for maximum concurrency. To give every transaction in this connection exclusive access to the data store, you can enable data store-level locking by setting the **LockLevel** attribute to 1. Doing so may improve performance for some applications. To cache Oracle tables, you must set row-level locking.

A connection can change the desired lock level at any time by calling the **ttLockLevel** built-in procedure (see "ttLockLevel" on page 248). Connections can also wait for unavailable locks by calling the **ttLockWait** built-in procedure (see "ttLockWait" on page 249). Different connections can coexist with different levels of locking, but the presence of even one connection doing data store-level locking leads to loss of concurrency. Logging to disk (see "Logging" on page 33) must be enabled to use row-level locking. To display a list of all locks on a particular data store you can use the TimesTen utility **ttXactAdmin**.

Setting Set LockLevel as follows:

C or Java programs or UNIX ODBC.INI file	LockLevel	**0**—Transactions access data store using row-level locking (default). **1**—Transactions access data store by acquiring an exclusive lock on the entire data store.
Windows ODBC Data Source Administrator	**DS-Level Locking** check box	**unchecked**—Transactions access data store using row-level locking (default). **checked**—Transactions access data store by acquiring an exclusive lock on the entire data store.

LockWait

Allows an application to configure the lock wait interval for the connection. The lock wait interval is the number of tenths of seconds to wait for a lock when there is contention on it. LockWait may be set to any value between 0 and 1,000,000 inclusive. The default value is 10 seconds.

Lock wait intervals are imprecise, and may be exceeded by up to 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.

A connection can change the lock wait interval at any time by calling the ttLockWait built-in procedure (see "ttLockWait" on page 249).

To display a list of all locks on a particular data store you can use the TimesTen utility **ttXactAdmin**.

Setting Set LockWait as follows:

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

MatchLogOpts

The first connection to a data store determines the type of logging, if any, that will be performed and whether the log files will be purged in the case of logging to disk. Any subsequent connection must specify the same values for the **Logging** and **LogPurge** attributes or an error will be generated. If a connection does not know the current state of these attributes, **MatchLogOpts** can be set so that the logging attributes will match.

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

Setting Set MatchLogOpts as follows:

C or Java programs or UNIX ODBC.INI file	MatchLogOpts	**0**—Values of **Logging** and **LogPurge** are used (default). **1**—Values of **Logging** and **LogPurge** are ignored. Instead, values match those of current connections.
Windows ODBC Data Source Administrator	**Match Log Opts** check box	**unchecked**—Values of **Logging** and **LogPurge** are used (default). **checked**—Values of **Logging** and **LogPurge** are ignored. Instead, values match those of current connections.

PermWarnThreshold

Indicates the threshold percentage at which TimesTen issues out-of-memory warnings for the permanent partition of the data store's memory. An application must call the built-in procedure ttWarnonLowMemory to receive out-of-memory warnings. The threshold also applies to SNMP warnings. See "ttWarnOnLowMemory" on page 329 and "Diagnostics through SNMP Traps" in *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

Setting Set PermWarnThreshold as follows:

C or Java programs or UNIX ODBC.INI file	PermWarn-Threshold	**p**—Percentage at which warning should be issued. Default is 90%
Windows ODBC Data Source Administrator	Low Memory Warning Thresholds for Permanent Data field	**p**—Percentage at which warning should be issued. Default is 90%

PrivateCommands

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

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

Setting

Set PrivateCommands as follows:

C or Java programs or UNIX ODBC.INI file	PrivateCommands	**0**—Commands are shared with other connections. (Default) **1**—Commands are not shared with any other connection.
Windows ODBC Data Source Administrator	**PrivateCommands** field	**0**—Commands are shared with other connections. (Default) **1**—Commands are not shared with any other connection.

Notes

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

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

PWDCrypt

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

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

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

Notes

If PWD and PWDCrypt are both supplied, the PWD value is used.

The PWDCrypt attribute cannot be used when access control is not enabled and **Authenticate**=1 or **GroupRestrict** is set. In this case, the native operating system authentication API is in use and expects the UID and PWD.

The PWD is not stored anywhere in the TimesTen system.

Setting

Set PWDCrypt as follows:

C or Java programs or UNIX ODBC.INI file	PWDCrypt	Enter the value generated by the **ttuser** utility.
Windows ODBC Data Source Administrator	PWDCrypt field	Enter the value generated by the **ttuser** utility.

SqlQueryTimeout

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

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

=Setting Set SqlQueryTimeout as follows:

C or Java programs or UNIX ODBC.INI file	SqlQueryTimeout	**n**—Time limit in seconds for which the data store should execute SQL queries.
Windows ODBC Data Source Administrator	**QueryTimeout (secs)** field	**n**—Time limit in seconds for which the data store should execute SQL queries.

TempWarnThreshold

Indicates the threshold percentage at which TimesTen issues out-of- memory warnings for the temporary partition of the data store's memory. An application must call the built-in procedure ttWarnonLowMemory to receive out-of-memory warnings. The threshold also applies to SNMP warnings. See "ttWarnOnLowMemory" on page 329 and "Diagnostics through SNMP Traps" in *Oracle TimesTen In-Memory Database Error Messages and SNMP Traps*.

Setting

Set TempWarnThreshold as follows:

C or Java programs or UNIX ODBC.INI file	TempWarnThreshold	**p**—Percentage at which warning should be issued. Default is 90%
Windows ODBC Data Source Administrator	Low Memory Warning Thresholds for Temporary Data field	**p**—Percentage at which warning should be issued. Default is 90%

UID

PWD

Access Control

When Access Control is enabled, a User Id and Password must be provided by users who are 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 more details on Access Control, see the *TimesTen Installation Guide*.

If an application wants to connect to a Client DSN whose corresponding Server DSN is defined with Authenticate=1, the application must provide a valid user name and password in the UID and PWD attributes, respectively.

The user must have rights to log on to the machine where the TimesTen Server is running. You may 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.

Note: The values for UID and PWD are stored in clear text in the `.odbc.ini` file on UNIX and in the Windows Registry on Windows. Any users who have access to the `.odbc.ini` file or Windows Registry can view the values for these attributes. You can use the PWDCrypt attribute to hide the cleartext value of the password. If PWD and PWDCrypt are both supplied, the PWD value is used. PWD is transmitted in encrypted form over the network from the TimesTen Client machine to the TimesTen Server machine.

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

Setting

Set UID and PWD as follows:

C or Java programs or UNIX ODBC.INI file	UID	Character string specifying the user ID.
C or Java programs or UNIX ODBC.INI file	PWD	Character string specifying the password that corresponds to the user ID.
Windows ODBC Data Source Administrator	**User ID** field	Character string specifying the user ID.

WaitForConnect

When an application requests a connection to a TimesTen data store and the connection is not possible (perhaps during concurrent loading/recovery of a data store), TimesTen normally waits for completion of the conflicting connection. In some cases, it can take some time for an application to connect to a data store. If the **WaitForConnect** attribute is off and the data store 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*.

Setting Set WaitForConnect as follows:

C or Java programs or UNIX ODBC.INI file	WaitForConnect	**0**—Does not wait if connection to data store fails. **1**—Waits until connection to data store is possible (default).
Windows ODBC Data Source Administrator	**Wait For Connect** check box	**unchecked**—Does not wait if connection to data store fails. **checked**—Waits until connection to data store is possible (default).

NLS general connection attributes

NLS connection attributes are set by each connection and persist for the duration of the connection. These attributes control the globalization behaviors of the data store. NLS general connection attributes are listed in the table "NLS general connection attributes" on page 12 and described in detail in this section.

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

ConnectionCharacterSet

This attribute is also available as a Client connection attribute.

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

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

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

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

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

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

This attribute accepts the same values used for the **DatabaseCharacterSet**. For a list of character set names that can be used as a value for this attribute, see "Supported Character Sets" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Setting Set ConnectionCharacterSet as follows:

C or Java programs or UNIX ODBC.INI file	Connection CharacterSet	The default value for ConnectionCharacterSet is US7ASCII, except when the data store character set is TIMESTEN8.
Windows ODBC Data Source Administrator	**Connection CharacterSet** list	The default value for ConnectionCharacterSet is US7ASCII, except when the data store character set is TIMESTEN8.

NLS_LENGTH_SEMANTICS

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

Setting Set NLS_LENGTH_SEMANTICS as follows:

C or Java programs or UNIX ODBC.INI file	NLS_LENGTH _SEMANTICS	Specify either BYTE (default) or CHAR
Windows ODBC Data Source Administrator	**NLS_LENGTH _SEMANTICS** list	Select either BYTE (default) or CHAR.

NLS_NCHAR_CONV_EXCP

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

Setting Set NLS_NCHAR_CONV_EXCP as follows:

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

NLS_SORT

The NLS_SORT attribute indicates which collating sequence to use for linguistic comparisons. It accepts the values listed in "Supported Linguistic Sorts" in the *Oracle TimesTen In-Memory Database Operations Guide*. All of these values may be modified to do case-insensitive sorts by appending _CI to the value. To perform accent-insensitive and case-insensitive sorts, append _AI to the value.

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

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

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

Only BINARY sort is supported with the TIMESTEN8 character set.

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

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

Setting Set NLS_SORT as follows:

C or Java programs or UNIX ODBC.INI file	NLS_SORT	Specify the linguistic sort sequence or BINARY (default).
Windows ODBC Data Source Administrator	**NLS_SORT** dropdown list	Specify the linguistic sort sequence or BINARY (default).

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 the table "Client connection attributes" on page 13 and described in detail in this section.

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

- Authenticate
- GroupRestrict
- ConnectionCharacterSet
- ConnectionName
- UID and PWD

TCP_Port

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

If you specify anything other than a logical server name for the TTC_Server attribute in the Client DSN definition, TimesTen Client assumes that the Server is running on the default TCP/IP port number. In such cases, if your Server is running on a port other than the default port, you must specify the port number in the ODBC connection string. For example, "TTC_SERVER=*server_host_name*; TTC_SERVER_DSN=*Server_DSN*;TCP_PORT=*server_port*" or "DSN=*Client_DSN*; TCP_Port=*server_port*".

Setting Set TCP_Port as follows:

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

TTC_Server

When connecting to a TimesTen data store using the TimesTen Client and Server, the TimesTen Client requires the specification of the network address and TCP port number of the machine running the TimesTen Server. As a convenience, TimesTen allows you to define a logical server name that contains the network address and port number pair. If you specify anything other than a logical server name for this attribute, TimesTen Client assumes that the Server is running on the default TCP/IP port number. In such cases, if your Server is running on a port other than the default port, you must specify the port number in the ODBC connection string. For example, "TTC_SERVER=*server_host_name*; TTC_SERVER_DSN=*Server_DSN*;TCP_PORT=*server_port*" or "DSN=*Client_DSN*; TCP_Port=*server_port*".

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

Note: TimesTen recommends that you specify a logical server name for the TTC_Server attribute. However, you can also specify a domain name server (DNS), host name or IP address for the TTC_Server attribute. If you do not use a logical server name and the TimesTen Server is listening on a non-default port number, you must provide the port number in the ODBC connection string. For example, "TTC_SERVER=*server_host_name*; TTC_SERVER_DSN=*Server_DSN*;TCP_PORT=*server_port*" or "DSN=*Client_DSN*; TCP_Port=*server_port*".

Setting

Set TTC_Server as follows:

C or Java programs or UNIX ODBC.INI file	TTC_Server	Character string specifying the logical server.
Windows ODBC Data Source Administrator	**Server Name or Network Address** field	Character string specifying the logical server.

TTC_Server_DSN

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

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

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

Set TTC_Server_DSN as follows:

C or Java programs or UNIX ODBC.INI file	TTC_Server_DSN	Character string specifying the DSN that resides on the Server machine.
Windows ODBC Data Source Administrator	**Server DSN field**	Character string specifying the DSN that resides on the Server machine.

TTC_Timeout

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

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

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

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

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

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

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

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

Setting Set TTC_Timeout as follows:

C or Java programs or UNIX ODBC.INI file	TTC_Timeout	A value between 0 and 99999 that represents the number of seconds that TimesTen Client waits for a connection before timing out. (The default value is 60.)
Windows ODBC Data Source Administrator	**Timeout Interval** field	A value between 0 and 99999 that represents the number of seconds that TimesTen Client waits for a connection before timing out. (The default value is 60.)

Cache Connect attributes

Cache Connect attributes are used only when you are using the Cache Connect to Oracle option to the Oracle TimesTen In-Memory Database. Cache Connect attributes are listed in the table "Cache Connect attributes" on page 12 and described in detail in this section.

OracleID

This attribute is used in conjunction with the Cache Connect component of TimesTen.

This attribute identifies the Service Name for the Oracle instance

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

Setting Set OracleID as follows:

C or Java programs or UNIX ODBC.INI file	OracleID	Character string specifying the Oracle ID.
Windows ODBC Data Source Administrator	**OracleID** field	Character string specifying the Oracle ID.

OraclePWD

This attribute is used in conjunction with the Cache Connect component of TimesTen. It identifies the password to the Oracle database that is being cached in TimesTen.

To cache Oracle tables, you must specify the OraclePWD attribute.

Access Control

If Access Control is enabled, you must also specify a value for the PWD attribute. When caching Oracle tables, PWD specifies the TimesTen password while OraclePWD specifies the Oracle password.

Setting

Set Oracle Password as follows:

C or Java programs or UNIX ODBC.INI file	OraclePWD	Character string specifying the Oracle Password.
Windows ODBC Data Source Administrator	**Oracle Password** field	Character string specifying the Oracle Password.

Note: The value for OraclePWD is stored in clear text in the `.odbc.ini` file on UNIX and in the Windows Registry on Windows. Any users who have access to the `.odbc.ini` file or Windows Registry can view the value for this attribute.

PassThrough

This attribute is used in conjunction with the Cache Connect component of TimesTen.

It specifies which SQL statements are executed only in the TimesTen data store and which SQL statements are passed through to the Oracle database. For more details on the TimesTen feature, Cache Connect, see the *TimesTen Cache Connect to Oracle Guide* and "CREATE CACHE GROUP" in the *Oracle TimesTen In-Memory Database SQL Reference Guide*.

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

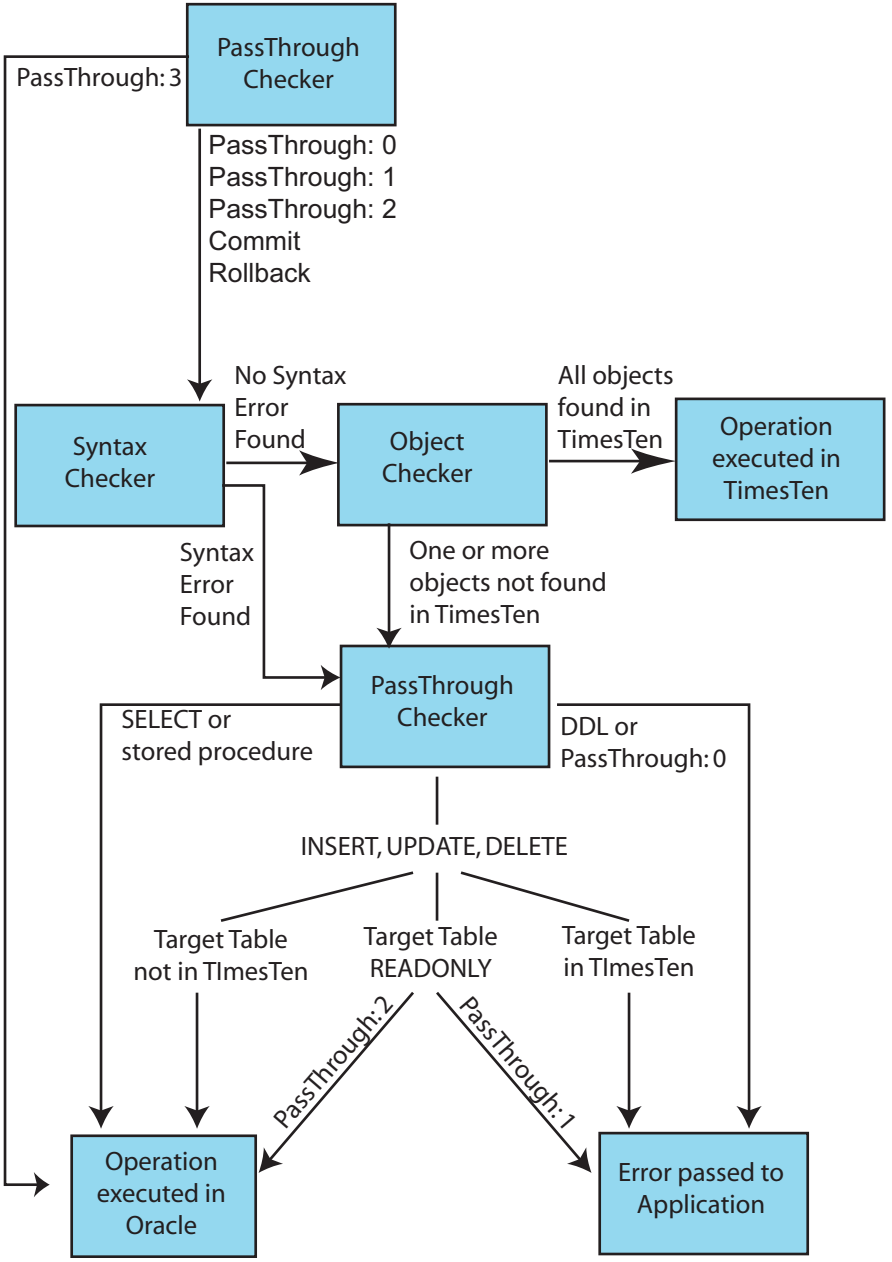
Setting	Set PassThrough as follows:	
C or Java programs or UNIX ODBC.INI file	PassThrough	**0** - SQL statements are executed only against TimesTen. **1** - Statements other than INSERT, DELETE or UPDATE and DDL are passed through if they generate a syntax error in TimesTen or if one or more tables referenced 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 will not be 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.
Windows ODBC Data Source Administrator	**PassThrough** List	**0** - SQL statements are executed only against TimesTen. **1** - Statements other than 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 will be passed through if the target table cannot be found in TimesTen. DDL statements will not be 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.

Certain restrictions need to be considered when using the PassThrough feature. They include:

- In the case that a SQL statement that uses TimesTen only syntax is passed through to the Oracle database, TimesTen returns an error message that indicates the syntax is not supported in Oracle.
- Execution of a prepared passthrough command assumes that the schema of dependent objects on Oracle have not changed after the prepare. It may cause unexpected results from Oracle or crash of the TimesTen system.
- TimesTen does not include a cache invalidation feature. TimesTen will not verify if the cached tables are up to date. When a query is syntactically correct in TimesTen and the cache contains all the tables referenced in the query, the query is executed in TimesTen regardless of whether the cache is up to date.
- The passthrough of Oracle INSERT, UPDATE and DELETE operations depends on the setting of the PassThrough attribute as described in the table above. TimesTen Cache Connect 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.

  The effects of a PassThrough INSERT, UPDATE and DELETE operation on a READONLY cache group are only seen after the transaction is committed and after the next AUTOREFRESH is completed.
- There is no mechanism to detect or block updates on an Oracle table that is cached in a TimesTen SYNCHRONOUS WRITETHROUGH cache group. Whether the updates are made by statements passed through the cache or from other non-cached Oracle applications the changes are never reflected in TimesTen.
- 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 will convert the C and SQL types and present the converted data and the SQL type code to TimesTen. TimesTen presents the information to OCI. The restrictions on supported data types are:
  - Length of the input binding values is restricted to 4000 for LONG type.
  - Input bindings for Oracle RAW and LONG RAW types are not supported in this release.
- Applications may get unexpected results from execution of a passthrough prepared command if the Oracle schema changes after prepare.

The effects of the PassThrough attribute are shown in the following figure and table.



Statement type	PassThrough Attribute Setting			
	0	1	2	3
Queries or procedure calls that use TimesTen syntax and only references TimesTen tables.	Executed in TimesTen	Executed in TimesTen	Executed in TimesTen	Passed through to Oracle
Queries or procedure calls that use either non-TimesTen syntax or references a table not in TimesTen.	Executed in TimesTen and returns an error.	Passed through to Oracle	Passed through to Oracle	Passed through to Oracle
INSERT, DELETE, MERGE or UPDATE statement where the target table is not in TimesTen.	Executed in TimesTen and returns an error.	Passed through to Oracle	Passed through to Oracle	Passed through to Oracle
INSERT, DELETE, MERGE or UPDATE statement where the target table exists in TimesTen.	Executed in TimesTen	Executed in TimesTen	Executed in TimesTen	Passed through to Oracle
INSERT, DELETE, MERGE or UPDATE statement where the target table is in a READONLY TimesTen cache group.	Executed in TimesTen and returns an error.	Executed in TimesTen and returns an error.	Passed through to Oracle	Passed through to Oracle
Any DDL statement.	Executed in TimesTen	Executed in TimesTen	Executed in TimesTen	Passed through to Oracle

RACCallback

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

For more details on the TimesTen feature, Cache Connect, see the *TimesTen Cache Connect to Oracle Guide* and "CREATE CACHE GROUP" in the *Oracle TimesTen In-Memory Database SQL Reference Guide*.

Setting Set RACCallback as follows:

C or Java programs or UNIX ODBC.INI file	RACCallback	**0** - Do not install TAF and FAN callbacks. **1** -Install the callbacks. (default)
Windows ODBC Data Source Administrator	**RACCallback** Checkbox	**checked** - Install the callbacks. (default) **unchecked** - Do not install TAF and FAN callbacks.

TransparentLoad

This attribute allows TimesTen to automatically load Oracle data into a TimesTen cache group when a SELECT query does not find data in the cache group tables.

For more details, see "Using transparent loading" in the *TimesTen Cache Connect to Oracle Guide* and "CREATE CACHE GROUP" in the *Oracle TimesTen In-Memory Database SQL Reference Guide*.

Setting	Set TransparentLoad as follows:	
C or Java programs or UNIX ODBC.INI file	TransparentLoad	**0** - Do not use transparent loading. (default) **1** - Run the SELECT statement in TimesTen without issuing error or warning messages. **2** - Return an error at compilation/execute time if the SELECT statement cannot use transparent load. The SELECT is executed based on the data available in TimesTen only.
Windows ODBC Data Source Administrator	**Transparent Load** pulldown menu	**0** - Do not use transparent loading. (default) **1** - Run the SELECT statement in TimesTen without issuing error or warning messages. **2** - Return an error at compilation/execute time if the SELECT statement cannot use transparent load. The SELECT will is executed based on the data available in TimesTen only.

2

Utilities

Overview

This chapter provides reference information and examples for TimesTen utilities.

The options for TimesTen utilities are generally not case sensitive, with the exception of single character options. You can use `-connstr` or `-connStr`. However `-v` and `-V` are each unique options.

All utilities return 0 for success and 1 if an error occurs.

Note: The utility name and options listed in this chapter are case-insensitive. They are described in mixed case to make the examples and syntax descriptions easier to read.

Access Control and utilities

If Access Control is turned on for your TimesTen instance, all utilities that require a password prompt for one.

If a UID attribute is given but no PWD attribute is given, either through a connection string or in the ODBCINI file for the specified DSN, TimesTen prompts for a password. When explicitly prompted, input is not displayed on the command line.

A password given on the command line, before TimesTen prompts for the password, is visible to the `ps` command, so use of the PWD attribute is not recommended in the first call to the utility. For example, the following command is not recommended:

```
% ttIsql –connStr “DSN=mydsn;UID=terry;PWD=secret”
```

When no UID attribute is given, the UID is assumed to be the user name identified by the operating system, and TimesTen does not prompt for a password.

Utilities list

Utility	See
ttAdmin	page 75
ttAdoptStores	page 81
ttBackup	page 83
ttBulkCp	page 86
ttCapture	page 102
ttCheck	page 104
ttDaemonAdmin	page 107
ttDaemonLog	page 109
ttDestroy	page 116
ttIsql	page 118
ttMigrate	page 136
ttmodinstall	page 154
ttRepAdmin	page 155
ttRestore	page 168
ttSchema	page 170
ttSize	page 174
ttStatus	page 176
ttSyslogCheck (UNIX)	page 179
ttTail	page 180
ttTraceMon	page 181
ttuser	page 184
ttVersion	page 185
ttXactAdmin	page 186
ttXactLog	page 196

ttAdmin

Description

Allows you to:

- Specify policies to automatically or manually load and unload data stores from RAM.
- Specify policies to automatically or manually start and stop replication agents for specified data stores.
- Start and stop TimesTen cache agents for caching data from Oracle tables. The cache agent is a process that handles Oracle database access on behalf of a TimesTen data store. It also handles the aging and autorefresh of the cache groups in the TimesTen data store. For the web pages of the Cache Connect Administrator to function correctly for a particular TimesTen data store, the corresponding cache agent must be started, since the agent is also responsible for accessing Oracle metadata for these web pages. Before using any cache features, you must start the cache agent.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility requires ADMIN privileges or data store object ownership. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttAdmin [-h | -help | -?]
ttAdmin [-V | -version]
ttAdmin [-ramPolicy always|manual|inUse [-ramGrace secs] ]
    [-ramLoad] [-ramUnload]
    [-repPolicy always|manual|norestart]
    [-repStart | -repStop]
    [[-cacheUidGet] |
     [-cacheUidPwdSet -cacheUid uid [-cachePwd pwd]] |
     [-cachePolicy always | manual] |
     [-cacheStart] |
     [-cacheStop [-stopTimeout seconds]]]
    [-query]
    {-connStr connection_string | DSN}
```

Options

ttAdmin has the options:

`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the data store to be administered.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-cachePolicy`	Defines the policy used to determine when the cache agent for the data store should run.
`-cacheStart`	Starts a cache agent for the data store.
`-cacheStop`	Stops a cache agent for the data store. You should not shut down the cache agent immediately after dropping or altering a cache group. Instead, wait for at least two minutes. Otherwise, the cache agent may not get a chance to clean up the Oracle objects that were used by the AUTOREFRESH feature.
`-cachePwd`	The password associated with the cache administration user ID that is used to manage AUTOREFRESH cache groups and ASYNCHRONOUS WRITETHROUGH cache groups. The cache administration user has extended privileges. See "Create Oracle users and set privileges" in the *TimesTen Cache Connect to Oracle 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 "Create Oracle users and set privileges" in the *TimesTen Cache Connect to Oracle Guide* for more details.
`-cacheUidGet`	Gets the current cache administration user ID for the specified data store.

`-cacheUidPwdSet`	Sets the cache administration user ID and password for the specified data store. Must be set in conjunction with the `-cacheUid` and `-cachePwd` options. Some things to consider are: • The cache administration user ID and password only need to be specified once for each new data store. • The cache administration user ID and password cannot be set while either the cache agent or the replication agent are running. • The cache administration user ID cannot be reset while there are ASYNCHRONOUS WRITETHROUGH cache groups or AUTOREFRESH cache groups (with a state that is not equal to OFF) on the data store. • The cache administration password can be changed at any time.
`-query`	Displays a summary of the policy settings for the named data store.
`-ramGrace` *`secs`*	Only effective if `-ramPolicy` is `inUse`. If non-zero, the data store is kept in RAM for *`secs`* seconds prior to being unloaded after the last application disconnects from the data store.
`-ramLoad`	Valid only when `-ramPolicy` is set to `manual`. Causes the data store to be loaded into RAM.
`-ramPolicy`	Defines the policy used to determine when the data store is loaded into system RAM. `always`—Specifies that the data store should remain in system RAM all the time. `manual`—Specifies that the data store is only to be loaded in system RAM when explicitly loaded by the user (using the `-ramLoad` option). `inUse`—(default) Specifies that the data store is only loaded in system RAM when in use (i.e., when applications are connected). The `-ramGrace` option may be used to modify the behavior of this policy. This option cannot be used with temporary data stores. TimesTen only allows a temporary data store to be loaded into RAM manually. Trying to set the policy generates a warning.
`-ramUnload`	Valid only when `-ramPolicy` is set to `manual`. Causes the data store to be unloaded from RAM.

`-repPolicy`	Defines the policy used to determine when the replication agent starts. `always`—Specifies that the agent should always be running for the data store. This option immediately starts the replication agent and when the daemon restarts the replication agent is restarted. `manual`—(default) Specifies that the replication agent must be manually started and stopped. `norestart` —Specifies that the replication agent for the data store is not to be restarted after a failure.
`-repStart`	Starts the data store's replication agent. TimesTen replication verifies that there is an entry in its tables for the local machine before starting the replication agent. If not, **ttAdmin** prints an error message saying that the replication agent did not start. The replication agent does not start if the data store does not participate in any replication scheme.
`-repStop`	Stops the data store's replication agent.
`-stopTimeout` *`seconds`*	Specifies that the TimesTen daemon should kill the cache agent if it doesn't stop within *seconds*. If set to 0, the daemon waits forever for the cache agent. The default value is 100 seconds.
`-V | -version`	Prints the release number of **ttAdmin** and exits.

Examples

A data store referred to by DSN `SalesData` is used by some very performance sensitive applications. So that applications do not have to wait for the data store to be loaded from disk into RAM, this data store must always remain in RAM. To achieve this, use:

```
ttAdmin -ramPolicy always SalesData
```

The SalesData data store is normally always resident in RAM. However, it is not being used at all today and should be loaded only when applications are connected to it. To change the RAM policy, use:

```
ttAdmin -ramPolicy inUse SalesData
```

To manually control whether the SalesData data store is loaded into RAM and to *load* it now, use:

```
ttAdmin -ramPolicy manual -ramLoad SalesData
```

To manually *unload* the SalesData data store from RAM, thus preventing any new applications from connecting to the data store, use:

```
ttAdmin -ramPolicy manual -ramUnload SalesData
```

A data store referred to by DSN `History` is not always in use. Permanently loading it into RAM unnecessarily uses memory. This data store 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 data store in RAM when applications are connected to it and to keep it in RAM for 5 minutes (300 seconds) after the last user disconnects. With this RAM policy, as long as applications are connected to the data store, the data store remains in RAM. To set this policy, use:

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

A data store referred to by DSN SalesData is used to cache Oracle data. Use the following **ttAdmin** command to start the cache agent for the SalesData DSN:

```
ttAdmin -cacheStart SalesData
```

You can also use the `-cachePolicy` option to ask the TimesTen data manager daemon to start the cache agent every time the data manager itself is started. Use:

```
ttAdmin -cachePolicy always SalesData
```

To turn off the automatic start of cache agent, use:

```
ttAdmin -cachePolicy manual SalesData
```

To set the cache administration user ID and password, the `-cacheUidPwdSet` flag in conjunction with the `-cacheUid` and `-cachePwd` options can be used with **ttAdmin**. For example, if the cache administration user ID and password on the data store SalesData should be scott and tiger respectively, use:

```
ttAdmin -cacheUidPwdSet -cacheUid scott -cachPwd tiger SalesData
```

To get the current cache administration user ID for the SalesData DSN, use:

```
ttAdmin -cacheUidGet SalesData
```

ttAdmin displays the following output:

```
Cache User Id: scott
RAM Residence Policy: inUse
Replication Agent Policy: manual
Replication Manually Started: False
Cache Agent Policy: manual
Cache Agent Mannualy Started: False
```

Notes

If TimesTen is installed as a user instance, and the user attempts to start the cache agent for a data store with a relative path, TimesTen looks for the data store relative to where it is running, and fails. Therefore, a relative path should not be used in this scenario. For example, on Windows, if you have specified the path for the data store as `DataStore=./dsn1` and attempt to start the cache agent with the command `ttAdmin -cacheStart dsn1`, the cache agent does not start because it looks for the data store in *`install_dir`*`\srv\dsn1`. For Unix it looks in a directory in `/var/TimesTen/`*`instance`*`/`.

When using AUTOREFRESH (automatic propagation from Oracle to TimesTen) or ASYNCHRONOUS WRITETHROUGH cache groups, you must specify the cache administration user ID and password. This user account is used to perform AUTOREFRESH and ASYNCHRONOUS WRITETHROUGH operations.

To load Oracle data, the TimesTen cache agent must be running. This requires that the ORACLE_HOME environment variable be set to the path of the Oracle installation. See the *TimesTen Cache Connect Guide* for more details. For details on other environment variables that you may want to set, see "Environment modifications" in the *Oracle TimesTen In-Memory Database Installation Guide*.

This utility is supported only for TimesTen Data Manager DSNs. It is not supported for TimesTen Client DSNs.

If **ttAdmin** is used with `-repStart` and it does not find a replication definition, the replication agent is not started and **ttAdmin** prints out an error message. For example:

```
$ ttAdmin -repStart rep1
*** [TimesTen][TimesTen 70.1 ODBC Driver][TimesTen]TT8191: Store not
involved in a replication scheme -- file "eeProc.c", lineno 9773,
procedure "RepAdmin()"
*** ODBC Error = S1000, TimesTen Error = 8191
```

If **ttAdmin** is used with the `-ramPolicy always` option, a persistent system connection is created on the data store. The `-ramPolicy always` option can only be specified on shared data stores.

The only `-ramPolicy` value supported for temporary data stores is the `-ramPolicy manual` option with the `-ramLoad` option specified at the same time.

If **ttAdmin** is used with `-repPolicy manual` (the default) or `-repPolicy always`, then the `-ramPolicy always` option should also be used. This ensures that the replication agent begins recovery after a failure as quickly as possible.

See Also

“ttStatus” on page 176
“ttCachePolicySet” on page 217
“ttCacheUidGet” on page 224
“ttCacheUidPwdSet” on page 225
“ttCacheStart” on page 222
“ttCacheStop” on page 223

ttAdoptStores

Description

On UNIX systems, moves data stores from a TimesTen instance to a new TimesTen instance that is of the same minor version. For example, you can move files from TimesTen 7.0.1 to TimesTen 7.0.2

The utility makes it easy to move to a new patch release and can be useful for testing a new release without uninstalling the old one. You can install the new version of TimesTen and move one or more data stores to the new release to test it without uninstalling the old one. Both utilities require that both instances be of the same minor version.

You must run the ttAdoptStores utility from the destination instance.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility must be run by the **TimesTen instance administrator** of the two instances. Alternatively, this utility can be run by the user root, whether or not Access Control is enabled for your TimesTen instance.

Syntax

```
ttadoptstores {-h | -help | -V}
ttadoptstores [-quiet] –dspath path
ttadoptstores [-quiet] –instpath path
```

Options

ttAdoptStores has the options:

`-dspath` *`path`*	Adopts a single data store. The *`path`* argument must be the path to the data store files (without any file extensions).

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-instpath` *`path`*	Adopts all data stores for an instance. The *`path`* argument must be the path to the daemon working directory (“infodir”). If any data stores are in use, the utility fails without making any modifications. No new connections to any data store 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 data store `/my/data/stores/ds`, use:

```
ttadoptstores -dspath /my/data/stores/ds
```

To adopt all of the data stores in the directory `/opt/TimesTen/instance1`, use:

```
ttadoptstores -instpath /opt/TimesTen/instance1
```

Note

You cannot adopt temporary data stores.

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 data store which can be restored at a later time using the **ttRestore** utility. If the data store is in use at the time of backup, it must be in *shared* mode to enable **ttBackup**. For an overview of the TimesTen backup and restore facility, see "To rename the owner of tables in a data store, use the ttMigrate utility. When restoring tables, you can use the -rename option to rename the owner of tables. Backing up and restoring a data store"in the *Oracle TimesTen In-Memory Database Operations Guide*.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility requires ADMIN privileges or data store object ownership. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttBackup [-h | -help | -?]
ttBackup [-V | -version]
ttBackup [-type backupType]
[-dir directory] [-fname fileprefix] [-force]
{-connStr connection_string | DSN}
```

Options

ttBackup has the options:

`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the data store 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 data store's ODBC definition.
`-force`	Forces the backup into the specified directory. If a backup already exists in that directory, **ttBackup** overwrites it. If this option is not specified **ttBackup** terminates with an end message without overwriting existing files.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-type` *`backupType`*	Specifies the type of backup to be performed. Valid values are: `fileFull` - Performs a full file backup to the backup path specified by the *directory* and *fileprefix* parameters (default). The resulting backup is not enabled for incremental backup. `fileFullEnable` - Performs a full file backup to the backup path specified by the *directory* and *fileprefix* parameters. The resulting backup is enabled for incremental backup. `fileIncremental` - Performs an incremental file backup to the backup path specified by the *directory* and *fileprefix* parameters, if that backup path contains an incremental-enabled backup of the data store. 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 data store. Otherwise, it performs a full file backup of the data store 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 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 data store to the `backup` directory in `/users/pat/TimesTen/backups`, use:

```
ttBackup -type fileFullEnable
         -dir /users/pat/TimesTen/backups FastIns
```

To copy the FastIns data store to the file `FastIns.back`, use:

```
ttBackup -type streamFull FastIns > FastIns.back
```

On Unix, to save the `FastIns` data store to a backup tape, use:

```
ttBackup -type streamFull FastIns | dd bs=64k of=/dev/rmt0
```

To backup a data store named `origDSN` to the directory `/users/rob/tmp` and restore it to the data store named `restoredDSN`, use:

```
ttBackup -type fileFull
         -dir /users/rob/tmp -fname restored origDSN
ttRestore -dir /users/rob/tmp -fname restored restoredDSN
```

Note

When an incremental backup has been enabled, TimesTen creates a backup hold in the log file. This hold can be seen using the **ttLogHolds** built-in procedure. The backup hold is used to determine which log records should be backed up

upon subsequent incremental backups. Only changes since the last incremental backup are updated. A side effect to creating the backup hold is that it prevents 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.

When **ttBackup** is used to back up a data store that has logging off, the backup file reflects only those transactions that committed before the most recent checkpoint. Transactions that committed after the most recent checkpoint are not reflected in the backup.

Up to one checkpoint and one backup may be active at the same time, with the following limitations:

- A backup never needs to wait for a checkpoint to complete.
- A backup may need to wait for another backup to complete.
- A checkpoint may need to wait for a backup to complete.

This utility is supported only where the TimesTen Data Manager is installed.

You cannot back up temporary data stores.

See Also

"ttBulkCp" on page 86
"ttMigrate" on page 136
"ttRestore" on page 168

ttBulkCp

Description

Copies data between TimesTen tables and ASCII files. **ttBulkCp** has two modes:

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

On UNIX, this utility is supported for TimesTen Data Manager DSNs. For Client DSNs, use the utility **ttBulkCpCS**.

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 verbosity] [-xp numRows | rollback]
[-Cc | -Cnone] [-tformat timeFormat]
[-tsformat timeStampFormat] [-dformat | -D dateFormat]
[-F firstRow] [-L lastRow] [-N ncharEncoding] [-Q 0|1]
[-S errLevel] {-connStr connection_string | DSN}
[owner.]tableName [dataFile ...]

ttBulkCp -o [-sc] [-v verbosity] [-A 0|1] [-Cc | -Cnone]
[-tformat timeFormat] [-tsformat timeStampFormat]
[-dformat | -D dateFormat]  [-N ncharEncoding]
[-noForceSerializable | -forceSerializable]
[-tsprec precision] [-Q 0|1]
{-connStr connection_string | DSN} [owner.]tableName
[dataFile]
```

Options **ttBulkCp** has the options:

`-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. **Note:** This option must be specified without a space between the option flag and the value.
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the data store to be copied.
`-D | -dformat` *`dateFormat`*	Sets the date format. Legal fixed values are described in "Date, time and timestamp values" on page 96. 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.

`-h`
`-help`
`-?`
	Prints a short usage message and exits.

`-forceSerializable` `-noForceSerializable`	The -forceSerializable option indicates that ttBulkCp should use serializable isolation regardless of the DSN or connection string settings. This is the default behavior. -noForceSerializable indicates that ttBulkCp should honor the isolation level in the DSN or connection string. If you specify the -noForceSerializable option and the DSN or connection string indicates a non-serializable isolation mode, a warning is included in the output: **Warning:** This output was produced using a non-serializable isolation level. It may therefore not reflect a transaction-consistent state of the table. For more information on isolation modes, see "Transaction isolation levels" in the *Oracle TimesTen In-Memory Database Operations Guide*.
`-helpfull`	Prints a longer usage message and exits.
`-i`	Selects copy-in mode.
`-m` *`maxErrors`*	Maximum number of errors to report. Default is 10; a few extra related errors may be reported. If 0, the utility only connects, then returns.
`-N` *`ncharEncoding`*	Specifies the input and output character encoding for NCHAR types. Valid values are LOCALE (default), UTF8 or ASCII. LOCALE sets the input and output encoding based on the character set setting of the user's OS session.
`-o`	Selects copy-out mode.
`owner`	Specifies the owner of the table to be saved or loaded. If *`owner`* is omitted, TimesTen looks for the table under the user's name and then under the user name SYS. This parameter is case-insensitive.
`-Q [0 | 1]`	Indicates whether character-string values should be enclosed in double quotes. `0`—indicates that strings should not be quoted. `1`—(the default) indicates that strings should be quoted. This option overrides the `QUOTES` file attribute.

`-s`*c*	Sets the default field-separator character to *c*. If no default field-separator is specified, a comma (,) is used. The `-s` option takes the values `\t` (tab) or any of the characters: `~ ! @ # % ^ & * ( ) = : ; | < > ? , /` This option overrides the `FSEP` file attribute. **Note:** This option must be specified without a space between the option flag and the value.
`tableName`	Specifies the name of the table to be saved or loaded. This parameter is case-insensitive.
`-tformat` *`timeFormat`*	Sets the time format. Legal values are defined in "Date, time and timestamp values" on page 96. The default value is `ODBC`. This option overrides the `TSFORMAT` file attribute. See also -D | -dformat and -tsformat.
`-tsformat` *`timestampFormat`*	Sets the timestamp format. Legal fixed values are described in "Date, time and timestamp values" on page 96. 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`*`verbosity`*	Sets the verbosity level. `0`—suppresses the summary. `1`—(the default) prints a summary of rows copied upon completion.

The following options can be used in copy-out mode only:

`-A [0 | 1]`	Indicates whether **ttBulkCp** should suppress attribute lines in the output file. `0`—(the default) **ttBulkCp** may write attribute lines into the output file. `1`—suppresses output of attribute lines.
`-tsprec` *`precision`*	When used with the -o option, truncates timestamp values to *`precision`*. **ttBulkCp** allows up to 6 digits in the fraction of a second field. Truncation may be necessary when copying timestamps using other RDBMS.

The following options can be used in copy-in mode only:

`-cp` *`numTrans`* `-cp final`	Sets the checkpoint policy for the copy in. A value of 0 indicates that **ttBulkCp** should never checkpoint the data store, even after the entire copy is complete. A non-zero value indicates that **ttBulkCp** should checkpoint the data store after every *`numTrans`* transactions, and again after the entire load is complete. A value of `final` indicates that **ttBulkCp** should checkpoint the data store only when the entire copy is complete. The default value is 0. Periodic checkpoints can only be enabled if periodic commits are also enabled. See the `-xp` option.

`-d error`
`-d warn`
`-d ignore`
	By default, **ttBulkCp** does not consider rows that are rejected because of constraint violations in a unique column or index to be errors.
	`-d error`– specifies that constraint violations should be considered errors. Duplicate rows are then counted against *`maxErrs`* (see `-m`) and placed into the error file (see `-e`).
	`-d warn`– specifies that **ttBulkCp** should copy the offending rows into the error file but should not count them as errors.
	`-d ignore`– (the default) specifies that **ttBulkCp** should silently ignore duplicate rows.
	Regardless of the setting of `-d`, the duplicate rows are not inserted into the table.

`-e` *`errFile`*	Indicates the name of the file where **ttBulkCp** should place information about rows that cannot be copied into the TimesTen table because of errors. These errors include parsing errors, type-conversion errors and constraint violations. The value of *`errFile`* defaults to `stderr`. The format of the error file is the same as the format of the input file (see "datafile format" on page 92), so it should be possible to correct the errors in the error file and use the corrected error file as an input file for a subsequent run of **ttBulkCp**.
`-F` *`firstRow`*	Indicates the number of the first row that should be copied. This option can be used (perhaps in conjunction 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 of 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.

`-S error`
`-S warn`
`-S ignore`
	By default, **ttBulkCp** issues an error when it encounters a value that exceeds its maximum scale. This error can be generated for a decimal value whose scale exceeds the maximum scale of its column or for a TIMESTAMP value with more than 6 decimal places of fractional seconds (i.e., sub-microsecond granularity).
	`-S error`—(the default) specifies that **ttBulkCp** should not insert a row containing a value that exceeds its maximum scale into the table and that it should place an error into the error file.
	`-S warn`—specifies that **ttBulkCp** should right-truncate the value to its maximum scale before inserting the row into the table and that it should place a warning into the error file.
	`-S ignore`—specifies that **ttBulkCp** should silently right-truncate the value to its maximum scale before inserting the row into the table.

`-t error`
`-t warn`
`-t ignore`
	By default, **ttBulkCp** issues an error when a CHAR, VARCHAR2, NCHAR, NVARCHAR2, BINARY or VARBINARY value is longer than its maximum column width.
	`-t error`—(the default) specifies that rows containing long string or BINARY attributes should not be inserted into the TimesTen table and that an error should be placed into the error file.
	`-t warn`—specifies that long string or BINARY attributes should be truncated to the maximum column length before being inserted into the table but that a warning should be placed into the error file.
	`-t ignore`—specifies that long string or BINARY attributes should be silently truncated to the maximum column length before being inserted into the table.

`-u error`
`-u warn`
`-u ignore`
	By default, **ttBulkCp** issues an error when a REAL, FLOAT or DOUBLE attribute underflows. Underflow occurs when a floating point number is so small that it is rounded to zero.
	`-u error`—(the default) specifies that rows containing a REAL, FLOAT or DOUBLE value that underflow should not be inserted into the TimesTen table and that an error should be placed into the error file.
	`-u warn`—specifies that 0.0 should be inserted for REAL, FLOAT or DOUBLE attributes that underflow, but that a warning should be placed into the error file.
	`-u ignore`—specifies that 0.0 should be silently inserted for REAL, FLOAT or DOUBLE attributes that underflow.

`-xp` *`numRows`* `-xp rollback`	Sets the transaction policy for the load. A value of 0 indicates that **ttBulkCp** should perform the entire load as a single transaction and should commit that transaction whether the load succeeds or fails. A value of `rollback` indicates that **ttBulkCp** should perform the entire load as a single transaction and should roll that transaction back if the load fails. A non-zero value indicates that **ttBulkCp** should commit after every *numRows* successful inserts. The default value is 1000. The `-xp` option can be used in conjunction with `-cp` option to enable periodic checkpointing of the data store.

datafile format

Every line of a **ttBulkCp** input file is one of the following: a blank line, a comment line, an attribute line or a data line.

- Blank lines are lines with no characters at all, including whitespace characters (space and tab). Blank lines are ignored by **ttBulkCp**.
- Comment lines begin with the comment character. The default comment character is #; this default can be overridden with the `-C` command-line option or the COMMENTCHAR file attribute (see "Attribute line format" on page 92). The comment character must be the first character on the line. Comment lines are ignored by **ttBulkCp**. Comments at the end of data lines are not supported.
- Attribute lines are used for setting file attributes, which control the formatting of the datafile. Attribute lines begin with the ten-character sequence `##ttBulkCp`. The full syntax for attribute lines is described in "Attribute line format" on page 92. Attribute lines can appear anywhere in the datafile.
- Data lines contain the rows of the table being copied. Data lines in the datafile and rows of the table correspond one-to-one; that is, each data line completely describes exactly one row. Each data line consists of a list of column values separated by the field separator character. The default field separator is a comma (,). This default can be overridden by the `-s` command-line option or the FSEP file attribute. The full syntax for data lines is described in "Data line format" on page 94.

Attribute line format

The format of an attribute line is:

`##ttBulkCp[:`*`attribute=value`*`]...`

Attribute lines always begin with the ten-character sequence `##ttBulkCp`, even if the comment character is not #. This sequence is followed by zero or more file attribute settings, each preceded by a colon.

Attribute settings remain in effect until the end of the input file or until they are changed by another attribute line in the same input file. The values of any file attributes that are omitted in an attribute line are left unchanged.

Command line options take precedence over the values in the file attributes that are supported by ttBulkCp. Those file attributes are:

- `VERSION`: Specifies the version of the file format used in the file, expressed as *major*`.`*minor*. The only supported version is 1.0.
- `FSEP`: Specifies the field separator character used in the file. The field separator can be set to `\t` (tab) or any of the characters:
  `~ ! @ # $ % ^ & * ( ) = : ; | < > ? , /`

  This option can be overridden with the `-s` command-line option.
- `QUOTES`: Indicates whether character string values in the file are enclosed in double quotes. The value can be 0, to indicate that strings are not quoted, or 1, to indicate that strings are quoted. This value can be overridden with the `-Q` option.
- `COMMENTCHAR`: Specifies the comment character used in the file. The comment character can be set to \t (tab) or any of the characters:
  `~ ! @ # $ % ^ & * ( ) = : ; | < > ? , /`

  The comment character can also be set to the value `none`, which disables the use of comments in the datafile.
- `DFORMAT`: Sets the date format. Legal values are described in "Date, time and timestamp values" on page 96. When a custom format is used, it should be enclosed in single quotes. This value can be overridden with the `-D/-dformat` command-line option. See also `TFORMAT` and `TSFORMAT`.
- `NCHARENCODING`: Indicates the encoding to be used for the NCHAR and NVARCHAR2 data types. The value may be either ASCII or UTF-8.
- `TFORMAT`: Indicates the time format. Legal values are described in "Date, time and timestamp values" on page 96. When a custom format is used, it should be enclosed in single quotes. This value can be overridden with the `-tformat` command-line option. See also `DFORMAT` and `TSFORMAT`.
- `TSFORMAT`: Sets the timestamp format. Legal values are described in "Date, time and timestamp values" on page 96. When a custom format is used, it should be enclosed in single quotes. This value can be overridden with the `-tsformat` command-line option. See also `DFORMAT` and `TFORMAT`.

Examples

The following header line sets the field separator character to `$` and disables quoting of character strings:

```
##ttBulkCp:FSEP=$:QUOTES=0
```

The following header line disables comments and sets the date format to the Oracle format:

```
##ttBulkCp:COMMENTCHAR=none:DFORMAT=Oracle
```

The following header line set the date format to a custom format:

```
##ttBulkCp:DFORMAT='Mon DD, YYYY'
```

Data line format

Data lines contain the row data of the table being copied. Each data line corresponds to a row of the table; rows cannot span input-file lines. A data line consists of a list of column values separated by the field separator character. Unnecessary whitespace characters should not be placed either before or after the field separator. The format of each value is determined by its type.

NULL values

NULL values can either be expressed as NULL (all capitals, no quotes) or as empty fields.

Character and unicode strings

CHAR, VARCHAR2, NCHAR, NVARCHAR2: If quoting of character strings is enabled (the default), then strings and characters must be enclosed in double quotes. If quoting of character strings is disabled, then any double-quote characters in the string are considered to be part of the string itself. **ttBulkCp** recognizes the following backslash-escapes inside a character string, regardless of whether quoting of strings is enabled:

- `\"` The double-quote character. If character-string quoting is enabled, then all double quote characters in the string must be escaped with a backslash. If character-string quoting is disabled, then it is permissible, but not necessary, to use the backslash.
- `\t` The tab character.
- `\n` The newline character.
- `\r` The carriage return character.
- `\\` The backslash character.
- *xyz* (CHAR and VARCHAR2 only) The character whose ASCII value is *xyz*, where *xyz* is a three-character octal number, as in `\033`.
- \u*xyzw* (NCHAR and NVARCHAR2 only) The character whose unicode value is *xyzw*, where *xyzw* is a four-digit hexadecimal number, as in`\ufe4a`. The \u*xyzw* notation is supported in both UTF-8 and ASCII encoding modes.

  In addition, any of the `~ ! @ # $ % ^ & * ( ) = : ; | < > ? , /` characters can be escaped with a backslash. Although it is unnecessary to escape these characters in most cases, doing so prevents them from being mistaken for a comment character or a field separator when character-string quoting is disabled.

If character-string quoting is enabled, the empty string (represented as `" "`) is distinct from NULL. If character-string quoting is disabled, then empty strings cannot be represented, as they cannot be distinguished from NULL.

For unicode strings, unicode characters encoded using UTF-8 multi-byte sequences are supported in the UTF-8 encoding mode only. If these sequences are used with the ASCII encoding mode, **ttBulkCp** interprets each byte in the sequence as a separate character.

For fixed-length CHAR and NCHAR fields, strings that are shorter than the field length are padded with blanks. For VARCHAR2 and NVARCHAR2 fields, the string is entered into TimesTen exactly as given in the datafile. Trailing blanks are neither added nor removed.

Binary values

BINARY, VARBINARY: If quoting of character strings is enabled (the default), binary values are delimited by curly braces (`{...}`). If quoting of character strings is disabled, then curly braces should not be used. Whether or not character-string quoting is enabled, 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 hex string is converted to a single BINARY byte, the hex 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 datafile.

If character-string quoting is enabled, a zero-length binary value (represented as `{ }`) is distinct from NULL. If character-string quoting is disabled, then zero-length binary values cannot be represented, as they cannot be distinguished from NULL.

Integer values

TINYINT, SMALLINT, INTEGER, BIGINT: Integer values consist of an optional sign followed by one or more digits. Integer values may not use E-notation. Examples:

```
-14          98765        +186
```

Floating-point values

REAL, FLOAT, DOUBLE: Floating-point values can be expressed with or without decimal points and may use E-notation. Examples:

```
3.1415      -0.00004      1.1e-3        5e3
 .56        -682          -.62E-4       170.
```

Fixed-point values

DECIMAL, NUMERIC: Decimal values can be expressed with or without decimal points. Decimal values may not use E-notation. Examples:

```
5           -19.5         -11           000
-.1234      45.           -57.0         0.8888
```

Inf, -Inf and NaN values

Inf, -Inf and Nan values: Infinity and Not a Number values can be represented as strings to represent the corresponding constant value (all are case in-sensitive):

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 in conjunction with AM/PM for copy-in mode.
`HH`	Hour (01 through 12). Must be used in conjunction with AM/PM for copy-in mode.
`MI`	Minute (00 through 59)
`SS`	Second (00 through 59)
`FF`	Fractional seconds. Six digits, unless overridden with the `-tsprec` option.
`FF`*n*	Fractional seconds (number of digits specified by *n*).
`+FF`	In copy-in mode, matches, optional decimal point plus one or more fractional seconds. In copy-out mode, same as `.FF`
`+FF`*n*	In copy-in mode, same as `+FF`. In copy-out mode, same as `.FF`*n*
`AM` `PM`	Meridian indicator without dots. In copy-in mode, this must be used with HH or HH12, but not HH24.

A.M. P.M.	Meridian indicator with dots. In copy-in mode, this must be used with HH or HH12, but not HH24.
DF	Current date format (can only be used in timestamp format).
TF	Current time format (can only be used in timestamp format).
- / ; :	Punctuation that are matched in copy-in mode or output in copy-out mode.
“text”	Text that is matched in input mode or output in copy-out mode.
*	Matches 0 or more whitespace characters (space or tab) in copy-in mode or outputs 1 space in copy-out mode.

Fixed, date, time and timestamp formats

For date values, the fixed formats are

Format	Description
ODBC	YYYY-MM-DD Example: 1997-01-03 (default value)
Oracle	DD-Mon-YYYY Example: 03-Jan-1997
SYBASE1	MM/DD/YYYY Example: 01/03/1997
SYBASE2	DD-MM-YYYY Example: 03-01-1997
SYBASE3	Mon*DD*YYYY Example: Jan 03 1997

For time values, the only fixed format is ODBC:

Format	Description
ODBC	HH24:MI:SS Example: 07:47:23

For timestamp values, the fixed formats are:

Format	Description
ODBC	YYYY-MM-DD*HH24:MI:SS+FF Example: 1997-01-03 07:47:23
Oracle	DD-Mon-YYYY*HH24:MI:SS+FF Example: 03-Jan-1997 07:47:23
SYBASE1	MM/DD/YYYY*HH24:MI:SS+FF Example: 01/03/1997 07:47:23
SYBASE2	DD-MM-YYYY*HH24:MI:SS+FF Example: 03-01-1997 07:47:23
SYBASE3	Mon*DD*YYYY*HH24:MI:SS+FF Example: Jan 03 199707:47:23

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. Note that 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 data store `mystore` into a file called `mytbl.dump`.

```
ttBulkCp -o DSN=mystore mytbl mytbl.dump
```

The following command loads the rows listed in file `mytbl.dump` into a table called `mytbl` on data store `mystore`, placing any error messages into the file `mytbl.err`.

```
ttBulkCp -i -e mytbl.err DSN=mystore mytbl mytbl.dump
```

The above command terminates after the first error occurs. To force the copy to continue until the end of the input file (or a fatal error), use `-m 0`, as in:

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

To ignore errors caused by constraint violations, use `-d ignore`:

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

Notes

ttBulkCp explicitly sets the **Overwrite** data store attribute to 0, to prevent accidental destruction of a data store. For more information, see "Overwrite" on page 36.

REAL, FLOAT or DOUBLE values may be rounded to zero when the floating point number is small.

When specifying date, time and timestamp formats, incomplete or redundant formats are not allowed in input mode. Specifiers that reference fields that are not present in the data type (for example a minute specifier in a date format) return errors in copy-out mode. In copy-in mode, the values of those specifiers are ignored.

The following caveats apply when disabling quoted strings in the **ttBulkCp** datafile:

- Empty strings and zero-length BINARY values cannot be expressed, as they cannot be distinguished from NULL.
- If the field separator character appears inside a character string, it must be escaped with a backslash or else it is treated as an actual field separator.
- If a data line begins with a character string and that string begins with the comment character, that character must be escaped with a backslash or else the line is treated as a comment. Setting the comment character to `none` can prevent this, as long as there are no actual comments in the file.

For UTF-8, NCHAR are converted to UTF-8 encoding and then output. UTF-8 input is converted to NCHAR.

For ASCII, those NCHAR values that correspond to ASCII characters are output as ASCII. For those NCHAR values outside of the ASCII range, the escaped Unicode format is used.

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

See Also
"ttBackup" on page 83
"ttMigrate" on page 136
"ttRestore" on page 168

ttCapture

Description

Captures information about the state of TimesTen at the time the command is used. This information may be useful in diagnosing problems. It is often the case that TimesTen Customer Support needs to 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.out.`*n* file in the directory from which you invoke the ttCapture utility. This utility collects general information that is usually relevant to support cases.

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

Syntax

```
ttCapture [-h | -help | -?]
ttCapture [-V | -version]
ttCapture [-noinstinfo] [-nosysinfo] [-stdout | -dest dir]
[-logdir dir] [dspath | DSN]
```

Options

ttBulkCp has the options:

`-dest` *dir*	Writes the output file to the designated directory. Must be used with the `-dsname` option.
DSN	Specifies an ODBC data source name of the data store to be checked.
dspath	The fully qualified name of the data store to be evaluated. This is not the DSN associated with the connection but the fully qualified data store path name associated with the data store as specified in the `DataStore=` parameter of the data store's ODBC definition. For example, for a data store consisting of files `/home/payroll/1997.ds0`, `/home/payroll/1997.ds1`, and several log files `/home/payroll/1997.log`*n*, *dspath* is `/home/payroll/1997`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-logdir` *`dir`*	Specifies the location of the log directory. Must be used with the `-dsname` option. If not specified, the log directory may not be available.
`-noinstinfo`	Do not capture any installation information.
`-nosysinfo`	Do not capture any system information.
`-stdout`	On UNIX systems, writes all output to `stdout`, instead of writing the output to a file.
`-V | -version`	Prints the release number of **ttCapture** and exits.

Examples To capture data on the test_db data store and write the data store 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 data store. You can specify a specific structure to be checked and a desired level of checking.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility requires ADMIN privileges or data store object ownership. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttCheck [-h | -help | -?]
ttCheck [-V | -version]

ttCheck [ [-blkDir] [-compHeap] [-header] [-heap] [-indexHeap] [-log]
[-permBlkDir] [-permHeap] [-tempBlkDir] [-tmpHeap]
[-tables tblName [...]] [-users userName [...]]
[-level levelNum] ] [...]
[-m maxErrors] [-f outFile] [-v verbosity]
{DSN | [-connstr] connStr | dspath}
```

Options

ttCheck has the options:

`-blkDir`	Checks all the block directories.
`-compHeap`	Checks the compilation heap structure.
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the data store to be checked.
`dspath`	The fully qualified name of the data store to be checked. This is not the DSN associated with the connection. It is the fully qualified data store path name associated with the data store as specified in the `DataStore=` parameter of the data store's ODBC definition. For example, for a data store consisting of files `/home/payroll/1997.ds0`, `/home/payroll/1997.ds1`, and several log files `/home/payroll/1997.log`*`n`*, *`dspath`* is `/home/payroll/1997`.
`-f` *`outFile`*	Specifies the output file name; defaults to `stdout`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-header`	Checks the content of the data store header.
`-heap`	Checks all heap structures.
`-indexHeap`	Checks the index heap structure.
`-level` *`levelNum`*	Can be used to indicate the level of checking for header, block directory, heap and table. Different structures can be checked using different levels in a same command. A level specification is applied to all structures specified to its left in the command string that do not already have a level specification. A level specification is applied to all structures if no structure is specified in the command string. 1—Checks sanity bytes and simple fields. For example, counts enums for validity in all high-level structures. 2—Does all checks in level 1, plus checks the validity of structures, referenced by fields in other structures. 3—Does all checks in level 2, plus checks each table row for column values. For example, checks valid VARCHAR2 and FLOAT sizes. 4—(the default) Does all checks in level 3, plus checks index/table mapping for each row and each index.
`-log`	Checks the log buffer.
`-m` *`maxErrors`*	Maximum number of errors to report. Default is 10; a few extra related errors may be reported. If 0, the utility only connects, then returns.
`-permBlkDir`	Checks the permanent partition block directory.
`-permHeap`	Checks the permanent heap structure.
`-tables` *`tblName`* `[...]`	Checks table(s) specified by *`tblName`*.
`-tempBlkDir`	Checks the temporary partition block directory.
`-tmpHeap`	Checks the temporary heap structure.
`-users` *`userName`* `[...]`	Checks tables belonging to the user(s) specified by *`userName`*.

`-V | -version`	Prints the release number of **ttCheck** and exits.
`-v` *`verbosity`*	0—no output (program's exit status indicates if an error was found). 1—(the default) enable error output only. 2—error output and a progress report.

Examples

To perform a check of all structures in the `test_db` data store, use:

```
ttCheck test_db
```

To perform a sanity check of all structures in the `test_db` data store, use:

```
ttCheck -level 1 test_db
```

To perform a check of all tables in the test_db data store, use:

```
ttCheck -tables test_db
```

To check the physical structures and row contents of all tables in the `test_db` data store, 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` data store, use:

```
ttCheck -heap -level 1 -tables -level 4 test_db
```

To check the physical structures and row contents of tables `tab1` and `tab2` in the `test_db` data store, 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 data store, 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 best way to ensure this is to set ExclAccess=1. If run on a shared data store and other transactions are active, **ttCheck** may return errors when the data store is in fact intact.

The **ttCheck** utility checks views in the same manner as other tables in a data store. The utility cannot verify that the contents of a view matches view query's result.

If no structures are specified, **ttCheck** checks all structures. No errors are returned if a specified table's name or user is not found.

This utility may take some time to run. Verbosity level 2 allows you to print a progress report.

This utility is supported only where the TimesTen Data Manager is installed.

ttDaemonAdmin

Description Starts and stops the TimesTen main daemon, Server and Webserver.

Access Control If Access Control is enabled for your instance of TimesTen, this utility requires instance administrator privileges. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttDaemonAdmin [-h | -help | -?]
ttDaemonAdmin [-V | -version]
ttDaemonAdmin [-force] {-start | -stop | -restart}
ttDaemonAdmin -startserver
ttDaemonAdmin [-force] -stopserver
ttDaemonAdmin [-startwebserver | -stopwebserver]
```

Options **ttDaemonAdmin** has the options:

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-force`	Starts or stops the TimesTen main daemon, even when warnings are returned or with -stopserver immediately stops the server processes.
`-restart`	Restarts the TimesTen main daemon.
`-start`	Starts the TimesTen main daemon.
`-startserver`	Starts the TimesTen Server daemon.
`-startwebserver`	Starts the TimesTen web server.
`-stop`	Stops the TimesTen main daemon.
`-stopserver`	Stops the TimesTen Server daemon. Without the `-force` option, client/server connections to TimesTen data stores 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 data stores are forcefully and immediately terminated, and then the server exits.
`-stopwebserver`	Stops the TimesTen web server.
`-V | -version`	Prints the release number of **ttDaemonAdmin** and exits.

Notes

Changes to the TimesTen Server and web server options are temporary. To permanently set or disable the TimesTen Server and web server options, you must change the options in the `ttendaemon.options` file.

The `-force` option should be used with caution, as it may leave data stores in a state where you must perform recovery procedures.

When you use this utility to restart the server, the TimesTen daemon reads the `ttendaemon.options` files to see if it has been changed since it was last read. If the file has been changed, TimesTen checks for the values of the options:

```
-server
-serverShmIpc
-serverShmSize
-noserverlog
```

See Also

For a description of all daemon options and instructions for changing the `ttendaemon.options` file, see "Managing TimesTen daemon options" in the "Working with the Oracle TimesTen Data Manager Daemon" chapter of *Oracle TimesTen In-Memory Database Operations Guide*.

ttDaemonLog

Description

TimesTen uses a *TimesTen daemon* (referred to as the *TimesTen Data Manager Service* on Windows) and other background processes, known as *subdaemons* and *agents*, to manage access to the data stores. The activities of these TimesTen processes are logged as events in the *TimesTen daemon log*. The **ttDaemonLog** utility allows you to control the type of events that are written to and fetched from the TimesTen daemon log.

Add note about ttsuport.log

There are two versions of the **ttDaemonLog** utility**:**

- ttDaemonLog for Windows
- ttDaemonLog for UNIX

ttDaemonLog for Windows

On Windows, TimesTen daemon log messages are written to the Windows Application Event Log. The **ttDaemonLog** utility controls which events are written to and fetched from the log and displayed to `stdout`.

Syntax

```
ttDaemonLog [-h | -help | -?]
ttDaemonLog [-V | -version]
ttdaemonlog [-show type] [-b | -r | -s] [-f] [-integer]
[-loglevel level [DSN | [-connstr] connStr]]
[-[no]logcomponent component [DSN | [-connstr] connStr]]
[-logreset] [-msg string] [-setquiet | -setverbose]
[-n computer]
```

Options

ttDaemonLog has the options:

`-b`	Prints all TimesTen-generated syslog entries.
`-f`	When the end of the log is reached, **ttDaemonLog** does not terminate but continues to execute, periodically polling the event log to retrieve and display additional TimesTen log records. This is useful, for example, for generating a display of log data that is updated in real time.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-integer`	Maximum number of lines at end of log to display. Defaults to 40 lines if `-f` is specified. If 0, no limit is set

`-logcomponent` *`component`* `-nologcomponent`	Specifies that the given component should be logged. If a DSN or connection string is specified as the component, logging applies only to the specified data store. If -nologcomponent is specified, no logging is done for the indicated component. Legal values for *`component`* are: ALL DAEMON DAEMONDBG WEBSERVER REPLICATION ORACLECONNECT
`-loglevel` *`level`*	Specifies that messages of level greater than or equal to the specified *`level`* should be logged. If a DSN or connection string is specified as the component, the option applies only to the specified data store.
`-logreset`	Resets event logging parameters.
`-msg string`	Insert *`string`* into the TimesTen daemon log.
`-n` *`computer`*	Displays the log from a different computer. Specify the Universal Naming Convention (UNC) name of the target computer.
`-r`	Print only the TimesTen Replication Agent log. (Same as `-show replication`.)
`-s`	Print only the TimesTen Server log. (Same as `-show server`.)
`-setverbose` `-setquiet`	Enable (`-setverbose`) or disable (`-setquiet`) TimesTen verbose daemon logging.
`-show` *`type`*	Limits output to the given type. Types are: **all** - Shows all. (default) **replication** - Shows only log entries for replication agents. (Same as `-r` option.) **ora** - Shows only log entries for Cache Connect agents. **server** - Shows only log entries for TimesTen Server. (Same as `-s` option.)
`-V | -version`	Prints the release number of **ttDaemonLog** and exits.

Examples

By default, the **ttDaemonLog** utility logs messages and errors from all of the TimesTen components. You can narrow the scope of what is written to the log by setting the `-nologcomponent` option. The `-nologcomponent` option can be applied to selected data stores or all data stores.

For example, to prevent messages and errors related to replication for all data stores from being written to the log, enter:

```
ttDaemonLog -nologcomponent replication
```

To prevent messages and errors related to replication for the *masterdsn* data store from being written to the log, enter:

```
ttDaemonLog -nologcomponent replication masterdsn
```

If, you want to prevent both replication and Cache Connect errors and messages from being written, enter:

```
ttDaemonLog -nologcomponent replication
ttDaemonLog -nologcomponent oracleconnect
```

If, after setting a `-nologcomponent` option, you want to re-enable writing errors for a component, you can use the `-logcomponent` option. For example, if after preventing both replication and Cache Connect errors from being logged, as shown in the example above, you want to re-enable logging of replication errors, enter:

```
ttDaemonLog -logcomponent replication
```

To re-enable logging for all TimesTen components, you can use the `-logreset` option:

```
ttDaemonLog -logreset
```

To display all of the output from the TimesTen daemon and server on your local machine, use:

```
ttDaemonLog
```

To display the log output from the host machine named, *backup1*, use:

```
ttDaemonLog -n backup1
```

To write the log output to the file `C:\TimesTen\logout\log1`, use:

```
ttDaemonLog -file C:\TimesTen\logout\log1
```

The TimesTen Server generates a message each time an application connects to or disconnects from a client DSN if these messages were specified to be generated during installation. To display just the server log messages, use:

```
ttDaemonLog -show server
```

To display just the replication agent messages, use:

```
ttDaemonLog -show replication
```

To display just the cache agent messages, use:

```
ttDaemonLog -show ora
```

To display all messages from the TimesTen processes, use:

```
ttDaemonLog -show all
```

To restore logging to its default “verbose” level, use the `-setverbose` option:

```
ttDaemonLog -setverbose
```

Notes

While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

To permanently set or disable verbose daemon logging, you must change the options in the `ttendaemon.options` file. See "Modifying informational messages" in the "Working with the Oracle TimesTen Data Manager Daemon" chapter of the *Oracle TimesTen In-Memory Database Operations Guide*.

ttDaemonLog for UNIX

Description

On UNIX, **ttDaemonLog** fetches all TimesTen events from the file generated by `syslogd(1)`. It displays all events to `stdout`.

The TimesTen daemon (`timestend`) records its event log via `syslog(2)`. The eventual disposition of the log information depends on the configuration of your `/etc/syslog.conf` file, which you can customize to log or ignore messages selectively. Messages can be logged into various files depending on the configuration of the file. These files can grow to be quite large. You should prune them periodically to conserve disk space.

Syntax

```
ttDaemonLog [-h | -help | -?]
ttDaemonLog [-V | -version]
ttDaemonLog [-show type] [-b | -r | -s] [-f] [-integer]
[-file filename] [-facility name]
[-loglevel level [DSN | [-connstr] connStr]]
[-[no]logcomponent component [DSN | [-connstr] connStr]]
[-logreset] [-msg string] [-setquiet | -setverbose]
```

Options

ttDaemonLog has the options:

`-b`	Prints all TimesTen generated syslog entries.
`-f`	When the end of the log is reached, **ttDaemonLog** does not terminate but continues to execute, periodically polling the event log to retrieve and display additional TimesTen log records. This is useful, for example, for generating a display of log data that is updated in real time.
`-facility` *`name`*	Specifies the syslog facility name being used.
`-file` *`filename`*	Specifies the file into which TimesTen is logging messages. If not specified, examine the system's `syslog` configuration to determine where TimesTen messages are being logged.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-integer`	Maximum number of lines at end of log to display. Defaults to 40 lines if `-f` is specified. If 0, no limit is set.

`-logcomponent` *`component`* `-nologcomponent`	Specifies that the given component should be logged, along with any other components that are already being logged. If a DSN or connection string is specified as the component, logging applies only to the specified data store. If `-nologcomponent` is specified, no logging is done for the indicated component. Legal values for *`component`* are: ALL DAEMON DAEMONDBG WEBSERVER REPLICATION ORACLECONNECT
`-loglevel` *`level`*	Specifies that messages of level greater than or equal to the specified *`level`* should be logged. If a DSN or connection string is specified as the component, the option applies only to the specified data store.
`-logreset`	Resets event logging parameters.
`-msg` *`string`*	Insert *`string`* into the TimesTen daemon log.
`-r`	Print only the TimesTen Replication Agent log. (Same as `-show replication`.)
`-s`	Prints the TimesTen Server log. (Same as `-show server`.)
`-setverbose` `-setquiet`	Enable (`-setverbose`) or disable (`-setquiet`) TimesTen verbose daemon logging.
`-show` *`type`*	Limits output to given type. Types are: **all** - Shows all. (default) **replication** - Shows only log entries for replication agents. (Same as `-r` option.) **ora** - Shows only log entries for Cache Connect agents. **server** - Shows only log entries for TimesTen Server. (Same as `-s` option.)
`-V | -version`	Prints the release number of **ttDaemonLog** and exits.

Examples

With the exception of the example with the `-n` option, all of the examples shown under "ttDaemonLog for Windows" on page 109 also apply to the UNIX version of **ttDaemonLog**. The following examples, show the use of some of the UNIX-specific options.

To write the log output to the file `/var/adm/syslog/syslog.log`, use:

```
ttDaemonLog -file /var/adm/syslog/syslog.log
```

To direct logging to the *local7* facility, use.

```
ttDaemonLog -facility local7
```

Notes

While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

To permanently set or disable verbose daemon logging, you must change the options in the `ttendaemon.options` file. See "Modifying informational messages"in the "Working with the Oracle TimesTen Data Manager Daemon"chapter of the *Oracle TimesTen In-Memory Database Operations Guide*.

For information about configuring `syslog`, see “Modifying informational messages” on page 65.

ttDestroy

Description

Destroys a data store including all checkpoint files, transaction logs and daemon catalog entries (though not the DSNs).

Access Control

If Access Control is enabled for your instance of TimesTen, this utility requires ADMIN privileges or data store object ownership. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttDestroy [-h | -help | -?]
ttDestroy [-V | -version]
ttDestroy [[-wait] [-timeout secs]] [-force]
{-connStr connection_string | DSN | dspath}
```

Options

ttDestroy has the options:

`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the data store to be destroyed.
`dspath`	The fully qualified name of the data store to be destroyed. This is not the DSN associated with the connection but the fully qualified data store path name associated with the data store as specified in the `DataStore=` parameter of the data store's ODBC definition. For example, for a data store consisting of files `/home/payroll/1997.ds0`, `/home/payroll/1997.ds1`, and several log files `/home/payroll/1997.log`*n*, *dspath* is `/home/payroll/1997`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-force`	Destroy even if files are from an incompatible version or a different instance of TimesTen.
`-timeout` *`seconds`*	Indicates the time in seconds that **ttDestroy** should wait. If no timeout value is supplied, TimesTen waits five seconds before retrying the destroy operation.

`-V | -version`	Prints the release number of **ttDestroy** and exits.
`-wait`	Causes **ttDestroy** to continually retry the destroy operation until it is successful, in those situations where the destroy fails due to some temporary condition, such as when the data store is in use.

Example

```
ttDestroy /users/pat/TimesTen/Daily/F112697
```

Notes

Using **ttDestroy** is the only way to delete a data store completely and safely. Do not remove data store checkpoint or log files manually.

This utility is supported only where the TimesTen Data Manager is installed.

ttDestroy does not perform cleanup of Oracle objects from AUTOREFRESH or AWT cache groups. If there are AUTOREFRESH or AWT cache groups in the data store, use the **ttCacheSqlGet** built-in procedure, specifying NULL for the cache group name, to generate Oracle SQL to perform cleanup after the data store has been destroyed.

ttIsql

Description

Runs SQL interactively from the command line. For a detailed description, use the `-helpfull` option.

On UNIX, this utility is supported for TimesTen Data Manager DSNs.

Syntax

```
ttIsql [-h | -help | -? | -helpcmds | -helpfull]
ttIsql [-V | -version]
ttIsql [-f inputFile] [-v verbosity]
[-e commands] [-interactive] [-N ncharEncoding] [-wait] {-
connStr connection_string | DSN}
ttIsql set attribute [value]
ttIsql show {all | attribute}
```

Options

ttIsql has the options:

`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the data store to be connected.
`-e` *`commands`*	Specifies a semicolon-separated list of **ttIsql** commands to execute on start up.
`-f` *`filename`*	Read SQL commands from *`filename`*.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-helpcmds`	Prints a short list of the interactive commands.
`-helpfull`	Prints a full description of the interactive commands.
`-interactive`	Forces interactive mode. This is useful when running from an emacs comint buffer.
`-N` *`ncharEncoding`*	Specifies the character encoding method for NCHAR output. Valid values are LOCALE or ASCII. LOCALE (the default) sets the output format to the locale-based setting. If no value is specified, TimesTen uses the system's native language characters.
`-V | -version`	Prints the release number of **ttIsql** and exits.

`-v verbosity`	Specifies the verbosity level. One of: 0—Shows error information only. If all commands succeed, there is no output. 1—The basic output generated by commands is displayed. 2—(the default) Same as level 1, plus it shows more detailed results of commands. At this level simplified SQL error and information messages are displayed. In addition, **ttIsql** commands that are read from an external file are echoed to the display. 3—Same as level 2, with more detailed error and information messages. 4—Same as level 3, plus complete error and information messages are displayed. Also displayed are messages about prepared commands, "success" messages for each command that succeeded and content of XLA records.
`-wait`	Waits until successful connect.

Commands

Also see the list of ttIsql **Set/Show attributes**.

Commands must be followed by a semi-colon (;). Boolean commands can accept the values "ON" and "OFF" in place of "1" and "0"

ttIsql has the commands:

`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. If the optional argument is not specified then information on all cache groups in the current data source is reported.

`cachesqlget` `[ASYNCHRONOUS_WRITETHROUGH | INCREMENTAL_AUTOREFRESH] [[`*`cache_group_owner`*`.] `*`cache_group_name`*`] {INSTALL | UNINSTALL} [`*`filename`*`]`	Generates an Oracle SQL*Plus compatible script for the installation or uninstallation of Oracle objects associated with a cache group using the incremental AUTOREFRESH or ASYNCHRONOUS WRITETHROUGH feature. If INSTALL is specified, the Oracle SQL statement to install the Oracle objects is generated. If UNINSTALL is specified, the Oracle SQL statement used to remove the Oracle objects is generated. If a cache group is not specified with UNINSTALL, a SQL statement to remove all Oracle objects in the AUTOREFRESH user's account is generated. If the optional *`filename`* argument is included, the generated SQL statement is saved to the specified external file. If the external file already exists, its contents are destroyed before writing to the file.
`clearhistory`	Clears the history buffer. Also see history and savehistory.
`clienttimeout` `[timeeout `*`seconds`*`]`	Sets the client timeout value in seconds for the current connection.
`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.
`commit`	Commits the current transaction (durably if **DurableCommits** is 1 for connection).
`commitdurable`	Commits the current transaction durably.
`compact`	Compacts the data store.

`connect [connection_string | DSN] [as connid]`	Connects to the data store 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.
`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 data store.
`describe [table_name_pattern | procedure_name_pattern | sql_statement|command_id | owner_name_pattern | *]`	Reports column definitions associated with a table name pattern, a procedure name pattern, or a prepared or unprepared SQL statement. For tables and cache groups, displays the aging policy if one exists and any CASCADE DELETE attributes. For column names, displays the INLINE properties of all variable length columns. If * is specified, reports the prepared statements for all connections.
`disconnect [all]`	Disconnects from the data store. If `all` is specified, disconnects and closes all connections. When disconnect finishes, the current connection is set to the reserved connection named "none."
`dssize [k|m]`	Prints data store size information in KB or MB. The default is KB.
`e: msg`	Echoes the specified messages, terminated by the end of the line. A semi-colon is not required to end the line. Messages are not echoed if verbosity is set to 0.

exec [*connect_id.*]*command_id*]	Executes the prepared command *command_id* on connection *connect_id*. If *command_id* is not specified, executes the most recent command.
execandfetch [*connect_id.*]*command_id*]	Executes and fetches all results from prepared command *command_id* on connection *connect_id*. If *command_id* is not specified, executes and fetches all results from the most recent command.
explain *sqlstmt*	Explains the plan for the specified SQL statement.
fetchall [*connect_id.*]*command_id*]	Fetches all results from prepared command *command_id* on connection *connect_id*. If *command_id* is not specified, fetches all results from the most recent command. The command must already have been executed using exec.
fetchone [*connect_id.*]*command_id*]	Fetches one result from prepared command *command_id* on connection *connect_id*. If *command_id* is not specified, fetches one result from the most recent command. The command must already have been executed using exec.
free [*connect_id.*]*command_id*]	Frees prepared command *command_id* on connection *connect_id*. If no command is specified, frees the most recent command.
help [*command* [*command ...*]| all | comments]	Prints brief or detailed help information for commands. If specific commands are given as arguments then detailed help for each command is printed. If you don't know the exact name of a command, try typing just a few characters that may be part of the command name. **ttIsql** searches and displays help for any commands that include the characters. If "all" is given as an argument then detailed help for all commands is printed. If "comments" is given as an argument then information on using **ttIsql** comments within scripts is printed. If no argument is given then brief help information for all commands is printed.

`history` `[-r] [num_commands]`	Lists previously executed commands. The *num_commands* parameter specifies the number of commands to list. If this parameter is omitted, the previous ten commands are listed by default. If the `-r` parameter is specified, commands are listed in reverse order. The history list stores up to 100 of the most recently executed commands. Use the clearhistory command to clear the history. Also see savehistory.
`host os_command`	Executes an operating system command. The command is executed in the same console as **ttIsql**. This command sets the environment variable TT_CONNSTR in the environment of the process it creates. The value of the variable is the connection string of the current connection.

`indexes`
`[[owner_name_pattern.]`
`table_name_pattern]`
	Lists indexes for the specified table or for all tables that match the given pattern.

`monitor`	Formats the contents of the MONITOR table for easy viewing.

`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.

`procedures [procedure_name_` `pattern]`	Reports built-in procedure names that match the given pattern. If you omit the optional argument, TimesTen reports all the procedures in the data source.
`quit`	Exits **ttIsql**.

`repschemes`
`[[scheme_owner_pattern.]`
`scheme_name_pattern]`
	Reports information on replication schemes defined in the currently connected data source. This information includes the attributes of all elements associated with the replication schemes. If the optional argument is not specified then information on all replication schemes defined in the current data source is reported.

`retryconnect [0|1]`	Disables(0) or enables(1) the wait for connection retry feature. If the connection retry feature is enabled then connection attempts to a data source that initially fail due to a temporary situation are retried until the connection attempt succeeds. For example, if data source recovery is in progress when attempting to connect, the connection retry feature causes the connect command to continue to attempt a connection until the recovery process is complete. If the optional argument is omitted then the connection retry feature is enabled by default.
`rollback`	Rolls back the current transaction. **AutoCommit** must be off.
`run` *`filename`*	Reads and executes SQL commands from *`filename`*. The run command can be nested up to five levels.
`savehistory` `[-a | -f]` *`outputfile`*	Writes the history buffer to the specified *output* file. Only command, no parameter values are saved in the output file. Therefore, a script may not be able to replay the history from the output file. If the *output* file already exists, you must specify either the `-a` or `-f` option. If `-a` is specified, the history is appended to the specified *output* file. If `-f` is specified, the history overwrites the contents of the specified *output* file. Also see clearhistory and history.

`sequences`
`[[`*`owner_name_pattern`*`.]`
`sequence_name_pattern``]`
	Reports the attributes of sequences matching the pattern. If you omit the optional argument, TimesTen reports all the sequences in the data source.

`set` *`attribute`* `[`*`value`*`]`	Sets the specified attribute to the specified value. If no value is specified, displays the current value of the specified attribute. For a description of accepted attributes, see **Set/Show attributes**.
`setjoinorder` *`tblNames`* `[...]`	Specifies the join order for the optimizer. **AutoCommit** must be off.

`setuseindex` *`index_name`*`, `*`correlation_name`*`, {0 | 1} [;...]`	Sets the index hint for the query optimizer.
`show [all |`*`attribute`*`]`	Displays the value for the specified data store attribute or displays all the attributes. For a description of accepted attributes, see **Set/Show attributes**.
`sleep [`*`n`*`]`	Suspends execution for *n* seconds. If *n* is not specified then execution is suspended for 1 second.
`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`*`.] `*`table_name`*`]`	Prints results of an ODBC call to SQLStatistics. `sqlstatistics` does not accept table name patterns.
`sqltables [[`*`owner_name_pattern`*`.] `*`table_name_pattern`*`]`	Prints results of a call to SQLTables. The pattern is a string containing an underscore (_) to match any single character or a percent sign (%) to match zero or more characters.
`statsclear [[`*`owner_name`*`.]`*`table_name`*`]`	Clears statistics for specified table (or all tables if no table is specified).
`statsestimate [[`*`owner_name`*`.]`*`table_name`*`] {`*`n`* `rows |` *`p`* `percent }`	Estimates statistics for specified table (or all tables if no table is specified).
`statsupdate [[`*`owner_name`*`.] `*`table_name`*`]`	Updates statistics for specified table (or all tables if no table is specified).
`tables [[`*`owner_name_pattern`*`.] `*`table_name_pattern`*`]`	Lists fully-qualified table, materialized view and view names. If the optional argument is omitted then ttIsql lists qualified names of all the tables, materialized views and views in the data store.
`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. If `use` fails to locate the connection id, the current connection is set to the reserved connection named “none.” (See “connect” on page 121.)
`version`	Reports version information.
`views [[owner_name_pattern.] table_name_pattern]`	Retrieves information about all views and materialized views defined in the data store.
`xladeletebookmark id`	Deletes a persistent XLA bookmark. If a bookmark to delete is not specified then the status of all current XLA bookmarks is reported. See “XLA Reference,” in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*. Requires ADMIN privilege or object ownership.

Set/Show attributes

Also see the list of ttIsql **Commands**. Some commands appear here as attributes of the **set** command. In that case, they can be used with or without the **set** command.

Boolean attributes can accept the values “ON” and “OFF” in place of “1” and “0”.

All commands must be followed by a semi-colon (;).

ttIsql set supports these attributes:

`all`	With show command only. Displays the setting of all the ttIsql commands. See example
`autocommit [1|0]`	Turns **AutoCommit** off and on. If no argument is given, displays the current setting.
`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.

`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.
`isolation [{READ_COMMITTED | 1}| {SERIALIZABLE | 0}]`	Sets isolation level. If no argument is supplied, displays the current value.
`multipleconnections [1 | 0]` `mc [1 | 0]`	Enables **ttIsql** to handle multiple connections. By default, **ttIsql** allows the user to have one open connection at a time. If the argument `1` (on) is specified the prompt is changed to include the current connection and all multipleconnection features are enabled. If no value is supplied, the command displays the value of the `multipleconnections` setting
`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.
`optfirstrow [1|0]`	Enables or disables First Row Optimization. If the optional argument is omitted, First Row Optimization is enabled.
`optprofile`	Prints the current optimizer flag settings and join order.

`passthrough [0|1|2|3]`	Sets the Cache Connect passthrough level for the current transaction. **AutoCommit** must be off to execute this command. **0** - executes all SQL statements against TimesTen. **1** - executes DML statements against Oracle if the target table cannot be found in TimesTen. **2**- similar to level1 plus executes any DML statements targeted at READONLY cache groups on Oracle. **3** - executes all SQL statements against Oracle. If no optional argument is supplied, the current setting is displayed. After the transaction, the passthrough value is reset to the value defined in the connection string or in the DSN or the default setting if no value was supplied to either.
`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.
`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.
`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.
`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.
`timing [1|0]`	Enables or disables printing of query timing.

`transparentload [0|1]`	Enables or disables transparent load (load on SELECT) for cache groups.
`tryhash [1|0]`	Enables or disables use of hash indexes by optimizer. **AutoCommit** must be off.
`trymaterialize [1|0]`	Enables or disables materialization by optimizer. **AutoCommit** must be off.
`trymergejoin [1|0]`	Enables or disables use of merge joins by optimizer. **AutoCommit** must be off.
`trynestedloopjoin [1|0]`	Enables or disables use of nested loop joins by optimizer. **AutoCommit** must be off.
`tryrowid [1|0]`	Enables or disables rowID scan hint.
`tryrowlocks [1|0]`	Enables or disables use of row-level locking by the optimizer. **AutoCommit** must be off.
`tryserial [1|0]`	Enables or disables use of serial scans by optimizer. **AutoCommit** must be off.
`trytbllocks [1|0]`	Enables or disables use of table-level locking by the optimizer. **AutoCommit** must be off.
`trytmptable [1|0]`	Enables or disables use of temporary tables by optimizer. **AutoCommit** must be off.
`trytmpttree [1|0]`	Enables or disables use of temporary T-tree (range) indexes by optimizer. **AutoCommit** must be off.
`tryttree [1|0]`	Enables or disables use of T-tree (range) indexes by optimizer. **AutoCommit** must be off.

`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.
`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 already turned 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.

Comment Syntax

The types of comment markers are:

`# [comment_text]`

`-- [comment_text]`

`/* [comment_text] */`

The comment should not span multiple lines. If a comment marker is encountered while processing a line, then the remainder of the line is ignored.

'`--`' at the beginning of a line is considered a SQL comment. The line is considered a comment and no part of the line is included in the processing of the SQL statement. A line that begins with '`--+`' is interpreted as a segment of a SQL statement.

The comment markers can work in the middle of a line.

Example:

```
monitor; /*this is a comment after a ttIsql command*/
```

Command History

ttIsql implements a csh-like command history.

Command Usage: `history [-r] [num_commands]`

Description: Lists previously executed commands. The *num_commands* parameter specifies the number of commands to list. If the `-r` parameter is specified, commands are listed in reverse order.

Command Usage: `! [command_id | command_string | !]`

Description: Executes a command in the history list. If a *command_id* argument is specified, the command in the history list associated with this ID is executed again. If the *command_string* argument is specified, the most recent command in the history list that begins with *command_string* is executed

again. If the `!` argument is specified then the most recently executed command is executed again.

Example: `"!!;"` -or- `"!10;"` -or- `"!con;"`

Also see the clearhistory, history, savehistory commands.

Command shortcuts

By default, **ttIsql** supports keystroke shortcuts when entering commands. To turn this feature off, use:

```
Command> set editline=0;
```

The bindings available are:

Keystroke	Action
Left Arrow	Moves the insertion point left (back).
Right Arrow	Moves the insertion point right (forward).
Up Arrow	Scroll to the command prior to the one being displayed. Places the cursor at the end of the line.
Down Arrow	Scrolls to a more recent command history item and puts the cursor at the end of the line.
Ctrl-A	Moves the insertion point to the beginning of the line.
Ctrl-E	Moves the insertion point to the end of the line.
Ctrl-K	“Kill” - Saves and erases the characters on the command line from the current position to the end of the line.
Ctrl-Y	“Yank”- Restores the characters previously saved and inserts them at the current insertion point.
Ctrl-F	Forward character - move forward one character. (See Right Arrow.)
Ctrl-B	Backward character - moved back one character. (See Left Arrow.)
Ctrl-P	Previous history. (See Up Arrow.)
Ctrl-N	Next history. (See Down Arrow.)

Parameters

With dynamic parameters, you are prompted for input for each parameter on a separate line. Values for parameters are specified the same way literals are specified in SQL.

SQL_TIMESTAMP columns can be added using dynamic parameters. (For example, values like ‘1998-09-08 12:1212’).

Parameter values must be terminated with a semicolon character.

The possible types of values that can be entered are:

- Numeric literals. Example: 1234.5
- Time, date or timestamp literals within single quotation marks. Examples:

  '12:30:00'
  '2000-10-29'
  '2000-10-29 12:30:00'
  '2000-10-29 12:30:00.123456'
- Unicode string literals within single quotation marks preceded by 'N'. Example: N'abc'
- A NULL value. Example: NULL
- The '*' character which indicates that the parameter input process should be aborted. Example: *
- The '?' character prints the parameter input help information. Example: ?

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.

Example 2.1 Execute commands from `ttIsql.inp`.

```
ttIsql -f ttIsql.inp
```

Example 2.2 Enable all output. Connect to DSN RunData and create the data store if it does not already exist.

```
ttIsql -v 4 -connStr “DSN=RunData;AutoCreate=1”
```

Example 2.3 Print the interactive commands.

```
ttIsql -helpcmds
```

Example 2.4 Print the full help text.

```
ttIsql -helpfull
```

Example 2.5 Display the setting for all ttIsql attributes:

```
Command> show all;

Connection independent attribute values:
```

```
columnlabels = 0 (OFF)
editline = 1 (ON)
multipleconnections = 0 (OFF)
ncharencoding = LOCALE
prompt = 'Command> '
verbosity = 2
vertical = 0 (OFF)

Connection specific attribute values:

autocommit = 1 (ON)
Connection String =
DSN=DS70;UID=joeuser;DataStore=/DS/
DS70;DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF
8;DRIVER=/opt/TimesTen/tt70/lib/
libtten.so;Authenticate=0;PermSize=20;TempSize=20;TypeMode=1;
isolation = READ_COMMITTED
Prefetch count = 5
Query timeout = 0 seconds (no timeout)

Current Optimizer Settings:
Scan: 1
Hash: 1
Ttree: 1
TmpHash: 1
TmpTtree: 1
TmpTable: 1
NestedLoop: 1
MergeJoin: 1
GenPlan: 0
TblLock: 1
RowLock: 1
Rowid: 1
FirstRow: 0
IndexedOr: 1
PassThrough: 0
BranchAndBound: 1
ForceCompile: 0
CrViewSemCheck: 1
ShowJoinOrder: 0
CrViewSemCheck: 1
TransparentLoad: 0
UseBoyerMooreStringSearch: 0

Current Join Order:
<>
Command>
```

Example 2.6 Prepare and exec an SQL statement.

```
ttIsql -connStr "DSN=RunData"

ttIsql (c) 1996-2000, TimesTen, Inc. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
All commands must end with a semi-colon.
(Default setting AutoCommit=1)
Command> prepare 1 SELECT * FROM my_table;
Command> exec 1;
Command> fetchall;
```

Example 2.7 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>
```

Example 2.8 When Access Control is enabled, 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.
All commands must end with a semi-colon.
(Default setting AutoCommit=1)
Command> CREATE USER terry IDENDTIFIED BY ‘secret’;
```

Example 2.9 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.
All commands must end with a semi-colon.
```

```
(Default setting AutoCommit=1)
Command> xladeletebookmark;

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> xladeletebookmark mybookmark;

Command> xladeletebookmark;
0 bookmarks found.
```

Notes

Multiple **ttIsql** commands are allowed per line separated by semicolons.

For UTF-8, NCHAR values are converted to UTF-8 encoding and then output.

For ASCII, those NCHAR values that correspond to ASCII characters are output as ASCII. For those NCHAR values outside of the ASCII range, the escaped Unicode format is used. For example:

```
U+3042 HIRAGANA LETTER A
```

is output as

```
Command> SELECT c1 FROM t1;
 < a\u3042 >
```

NCHAR parameters must be entered as ASCII N-quoted literals:

```
Command> prepare SELECT * FROM t1 WHERE c1 = ?;
Command> exec;

All values for Parameters must end with a semi-colon character.
Type '?;'  for help on entering parameter values.
Type '*;' to abort the parameter entry process.

Enter Parameter  1> N'XY';
```

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

ttMigrate

Description

Performs one of these operations:

- Saves a migrate object from a TimesTen data store into a binary datafile.
- Restores the migrate object from the binary datafile into a TimesTen data store.
- Examines the contents of a binary datafile created by this utility.

Migrate objects include:

- Tables
- Cache group definitions
- Views and materialized views
- Sequences
- Replication schemes.

The **ttMigrate** utility is used when upgrading major release versions of TimesTen, since major revisions are not compatible. For an example, see the *Oracle TimesTen In-Memory Database Installation Guide*.

Binary files produced by this utility are platform-dependent. For example a binary file produced on Windows must be restored on Windows. Use the **ttBulkCp** utility to copy data between platforms.

The **ttMigrate** utility can be used to copy data between bit-levels within the same architecture. For example, it can be used to move data from a 32-bit Solaris system to a 64-bit Solaris system. The `-noRepUpgrade` option must be used when changing bit-levels and the data store should not be involved in a replication scheme, in this case.

On UNIX, this utility is supported for TimesTen Data Manager DSNs. For TimesTen Client DSNs, use the utility **ttMigrateCS**.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility requires SELECT privileges or table ownership to migrate tables and ADMIN privileges to migrate an entire data store. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

This utility does not migrate any granted privileges, regardless of whether Access Control is enabled on the new or old instance of TimesTen. You must explicitly create users and grant privileges on the migrated data store if Access Control is enabled on your new TimesTen instance.

Syntax

```
ttMigrate [-h | -help | -?]
ttMigrate [-V | -version]
```

To create or append a binary datafile, use:

```
ttMigrate  {-a | -c} [-vverbosity] [-nf] [-nr] [-fixNaN]
[-saveAsCharset charset]
[-repUpgrade | -noRepUpgrade
[-convertTypesToOra | -convertTypestoTT]]
{-connStr connection_string | DSN} dataFile
[objectOwner.]objectName
```

To restore a data store from a binary datafile created by this utility, use:

```
ttMigrate -r -noRepUpgrade [-inline rule] [-vverbosity] [-fixNaN]
[-nf] [-nr] [-C chkPtFreq] [-rename oldOwner:newOwner]
[-localhost hostName] [-n noCharsetConversion]
[-cacheUid uid [-cachePwd pwd]] [-convertCGtypes]
[-convertTypesToOra | -convertTypesToTT]
-connStr connection_string | DSN} dataFile
[objectOwner.]objectName
```

or

```
ttMigrate -r -repUpgrade [-vverbosity] [-fixNaN] [-nf] [-nr]
[-C chkPtFreq] [-rename oldOwner:newOwner]
[-localhost hostName] [-noCharsetConversion]
[-cacheUid uid [-cachePwd pwd]] [-convertCGtypes]
{-connStr connection_string | DSN} dataFile
[objectOwner.]objectName
```

To list or display the contents of a binary datafile created by this utility, use:

```
ttMigrate {-l | -L | -d | -D} dataFile [owner.name ...]
```

Options

Note: The append (-a) or create (-c) modes, the list (-l/-L) or describe (-d/-D) modes and the restore (-r) modes are exclusive of each other. You cannot specify any of these options on the same line as any other of these options

ttMigrate has the options:.

`-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 143 for more details.
`-c`	Create mode: Creates an original binary datafile. See "Create mode (-c) and Append mode (-a)" on page 143 for more details.

`-cacheUid`	The cache administration user ID to use when restoring AUTOREFRESH and ASYNCHRONOUS WRITETHROUGH cache groups.
`-cachePwd`	The cache administration password to use when restoring AUTOREFRESH and ASYNCHRONOUS WRITETHROUGH cache groups. If the cache administration user ID is provided on the command line but the cache administration password is not, then **ttMigrate** prompts for the password.
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`-convertTypesToOra` `-convertTypesToTT`	Converts TimesTen data types to Oracle data types or Oracle data types to TimesTen data types. These options require the `-noRepUpgrade` option. In TimesTen 7.0 the default type mode is ORACLE type mode. The -convertTypesToOra is useful when migrating older data stores into TimesTen 7.0. The `-convertTypesToTT` option is useful to allow backward migration into a release that does not support Oracle types. These options apply to all table types except materialized views. Table types include: regular, cached, global and temporary tables. **Note:** These options do not convert type names used in CAST expressions in the definitions of views or materialized views. Views or materialized views that use CAST statements may therefore fail to migrate between databases with different type modes. In addition these data type conversion procedures re not guaranteed to be reversible. Converting data from TimesTen types to Oracle types and back can result in columns and data that are different from the original in some cases. See also -convertCGTypes, "TimesTen to Oracle data type conversions" on page 148 and "Oracle to TimesTen data type conversions" on page 149."
`-d`	Selects Describe mode. Displays a short description of the objects in the datafile. See "Describe mode (-d)" on page 147 for more details.

`-D`	Selects Long-describe mode. Displays a full description of the objects in the datafile. See "Long-describe mode (-D)" on page 147 for more details.
`dataFile`	The path name of the datafile to which migrate objects are to be saved or from which migrate objects are to be restored.
`DSN`	Specifies an ODBC data source name of the data store to be migrated.
-fixNaN	Converts all NaN, Inf and -Inf values found in migrate objects to 0.0. This is useful for migrating data into releases of TimesTen that do not support the NaN, Inf and -Inf values.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-inline` *`rule`*	Indicates the *`rule`* to be used for converting variable-length columns to INLINE in restore mode. The value for *`rule`* is one of: `preserve` - **ttMigrate** preserves the original INLINE attribute of each column. This is the default, and it is required if `-repUpgrade` is used. `dsDefault` - **ttMigrate** uses the data store's default rule for setting the INLINE attribute of restored columns. *`maxlen`* - **ttMigrate** restores as INLINE all variable-length columns with length <= *`maxlen`* and restores as NOT INLINE all variable-length columns with length greater than *`maxlen`*. If *`maxlen`* is 0 then all columns are restored as NOT INLINE. When this option is used during migration, the data store should not be replicated with data stores from a TimesTen version that does not support INLINE columns. INLINE variable-length columns can not successfully be replicated to NOT INLINE columns.
`-l`	Selects List mode. Lists the names of data store objects in the specified datafile. See "List mode (-l) and Long-list mode (-L)" on page 146 for more details.
`-L`	Selects Long-list mode. Lists the names of data store objects in the specified datafile and other details about the data store objects. See "List mode (-l) and Long-list mode (-L)" on page 146 for more details.

`-r`	Selects Restore mode. Restores a data store from a binary datafile created by this utility. See "Restore mode (-r)" on page 145 for more details.
`name`	The name of the data store 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.
`-noRepUpgrade`	Do not ensure that saved/restored tables are compatible for replication with the original tables. Use of this option may cause the restored tables to be slightly more compact and slightly faster to access than otherwise. This option is ignored when the `-a` option is given. This option should be used with care, however, as replication between the original tables and the restored tables may not work properly.
`owner`	The owner of a migrate object.

`-rename` *`oldOwner:newOwner`*	Restores all data store objects previously owned by *oldOwner* to be owned by *newOwner*. You cannot change ownership of the TimesTen system tables (those tables owned by TTREP or SYS). This option should be used with some caution. When using the `-rename` option, be aware that: • If the new owner is not a user that has been given access to the data store in an Access Control TimesTen instance, **ttMigrate** still changes the owner and does not return an error or warning. • If the old owner name does not exist, a warning is printed at the end of the **ttMigrate** operation that no object with the specified owner was renamed. • If creating a data store object with the new owner returns an error, ttMigrate prints an error and continues restoring the rest of the data store objects. • You cannot change the owner of tables owned by TTREP or SYS. • If you change an owner name such that the table's owner and table name are now identical to another table in the data store, then the table is not restored correctly. • If you change the owner name of a table referenced by a view, you may need to drop and re-create the view. • If you change the owner of a replicated table, you must change the table's owner on all replication peers in order to replicate updates successfully.

`-repUpgrade`	Ensures that the saved/restored tables are compatible for replication with the original tables when restoring tables for replication upgrade. This option is ignored when the `-c or -a` options are given. The **ttMigrate** utility in restore mode sets `-repUpgrade` by default. When restoring data from a **ttMigrate** datafile, if the file was created by a version of **ttMigrate** that does not support the `-repUpgrade` option, or if you explicitly specified `-noRepUpgrade` when you created the file (with a **ttMigrate** version prior to 5.1), you may encounter an error that indicates that a migration object could not be created with "replication compatibility." This option is set by default. If you use a version of **ttMigrate** that does not support the `-repUpgrade` option, or you explicitly specify `-noRepUpgrade` when creating a migration file, you may encounter an error that indicates that a migration object could not be created with "replication compatibility." You must either obtain a version of **ttMigrate** that supports the `-repUpgrade` option for the TimesTen release from which you are migrating, or, if supported, specify `-noRepUpgrade` when migrating files into a version of TimesTen that sets `-repUpgrade` by default.
`-saveAsCharset` *`charset`*	Allows you to save an object in a character set other than the database character set. When saving an object, ttMigrate stores it in the database character set by default.
`-v`*`verbosity`*	Specifies the verbosity level for messages printed when **ttMigrate** saves or restores a data store. One of: 0—shows errors and warnings only. 1—prints the name of each table as it is saved or restored. 2—prints the name of each table or index as it is saved or restored. 3—(the default) prints the name of each table or index as it is saved or restored and prints a dot (.) for each 10,000 rows saved or restored. -v is ignored in List, Long-list, Describe and Long-describe modes.
`-V | -version`	Prints the release number of **ttMigrate** and exits.
The following options are available in Restore mode (with -r) only:	

`-c chkPtFreq`	Specifies that **ttMigrate** should checkpoint the data store after restoring every *chkPtFreq* megabytes of data. A value of zero (the default) specifies that **ttMigrate** should never checkpoint the data store.
`-convertCGTypes`	Determines the best type mapping from the underlying Oracle tables to TimesTen cached tables using: • The types of the columns in the Oracle tables • The types of the columns stored in the migration file and • The TimesTen-to-Oracle type mapping rules described above. If this option is specified with either the `-convertTypesToOra` or the `-convertTypesToTT` option, this option takes precedence for cached tables. This option does not impact non-cached tables.
`-localhost hostName`	Can be used to explicitly identify the name or IP address of the local host when restoring replicated tables.
`-noCharsetConversion`	Restores data in the database character set, not the tagged character set. See also -saveAsCharset. This option may be useful for legacy TimesTen users who may have migrated pre-7.0 data into a 7.0 or later release of TimesTen as TIMESTEN8 or another character set such as WE8ISO8895P1, when the data is actually in another character set. If, at a later time you wish to have that data interpreted according to its actual character set, use this option to migrate the data into a data store that uses the data's actual character set with no character set conversion.

Modes

Create mode (-c) and Append mode (-a)

In Create mode, **ttMigrate** saves migrate objects from a TimesTen data store into a new binary datafile. If the datafile does not exist, **ttMigrate** creates it. Otherwise, **ttMigrate** overwrites the existing file, destroying its contents.

The datafile format used by **ttMigrate** is independent of any release of TimesTen, so it is possible to use **ttMigrate** to migrate data from one TimesTen release to another.

You can save migrate objects from different releases of TimesTen in the same datafile.

In Append mode, **ttMigrate** appends migrate objects from a TimesTen data store to an existing datafile. If the datafile does not exist, **ttMigrate** creates it.

For each ordinary (non-cached) table, **ttMigrate** saves:

- The table description: the name and type of each of the table's columns, including primary key and nullability information.
- The table's index definitions: the name of each index and the columns contained in the index. The actual contents of the index are not saved; **ttMigrate** only saves the information needed to rebuild the index when the table is restored.
- The table's foreign key definitions. You can disable the saving of foreign key definitions using the `-nf` option.
- The rows of the table. You can disable the saving of rows using the `-nr` option.

For each cache group, **ttMigrate** saves the following:

- The cache group definition: the cache group owner and name, the names of all tables in the cache group and any relevant cache group settings, such as the cache group duration.

Note: After **ttMigrate** has been used to restore a data store, all autorefresh cache groups in the restored data store have AUTOREFRESH STATE set to OFF, no matter how it was set on the source data store. Reset AUTOREFRESH STATE to ON by using the ALTER CACHE GROUP statement.

- All the cached tables in the cache group: the table name, column information, table attributes (propagate or read-only), WHERE clause, if any, foreign key definitions and index definitions.

For each view, **ttMigrate** saves the following:

- All of 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.

If there are any replication schemes defined, **ttMigrate** saves all the of the TTREP tables containing the replication schemes. Replication schemes should have names that are unique from all other data store objects. It is not possible to migrate a replication scheme with the same name as any other database object.

Note: The **ttMigrate** utility does not save the rows of a cached table into the datafile, even if you have not specified the `-nr` option. The foreign key definitions of cached tables are always saved, regardless of the use of the `-nf` option, as they are needed to maintain the integrity of the cache group.

By default, **ttMigrate** saves all data store objects in the data store to the datafile, including tables, views, cache groups, sequences and replication schemes. Alternatively, you can give a list of data store objects to be saved on the command line, with the exception of 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 data store objects that match any of the given patterns. Names do not need to be fully qualified: If a name is given with no owner, **ttMigrate** saves all data store 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 data store objects from a datafile into a TimesTen data store.

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. You can use the `-rename` option to restore tables with a new owner name.
- 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. You can use the `-rename` option to restore cache groups with a new owner 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 read-only), and the WHERE clause, if any. You can use the `-rename` option to restore tables with a new owner name.
- The foreign key definitions of the cached tables.
- All of the indexes on the cached tables.

Note: The **ttMigrate** utility does not restore the rows of cached tables, even if you have not specified the `-nr` option. The foreign key definitions of the cached

tables are always restored, regardless of the use of the `-nf` option, as they are needed to maintain the integrity of the cache group.

By default, the `-repUpgrade` option is set during restore.

By default, **ttMigrate** restores all tables and cache groups in the datafile. Alternatively, you can list specific tables and cache groups to be restored on the command line. The names in this list must be fully qualified and cannot use wildcard characters.

You cannot restore cached tables independently of their cache groups. If you list a cached table on the command line without also listing the corresponding cache group, then **ttMigrate** issues an error.

Use the `-v` option to control the information that **ttMigrate** prints while the restoration is in progress.

The `-inline` option may be used to control whether variable length columns are restored as INLINE or NOT INLINE. See "Type specifications" in *Oracle TimesTen In-Memory Database SQL Reference Guide*. In the default mode, `-inline preserve`, **ttMigrate** restores all variable-length columns with the same INLINE or NOT INLINE setting with which they were saved. In the other two modes, `-inline dsDefault` 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 `-inline dsDefault`, this threshold is the default automatic INLINE length for a TimesTen data store. 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 data store objects in the specified datafile, including cached tables and the replication scheme TTREP tables.

In Long-list mode, **ttMigrate** lists the names of data store objects in the datafile, including cached tables and the replication scheme TTREP tables, along with the number of rows in each table and the index definitions for each table, the query defining each view and the specifications for each sequence.

By default, **ttMigrate** lists the replication scheme name as well as all of the data store objects in the file. Alternatively you can provide a list of names of data store 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 data store 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 as well as all of the TTREP replication scheme tables in the same manner as user tables.

By default, **ttMigrate** describes all of the data store objects in the file. Alternatively, you can provide a list of names of data store 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 data store 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 of the values used to define the sequence, as well as its current value.

For replication schemes, **ttMigrate** lists all of the TTREP replication scheme tables in the same manner as user tables.

By default, **ttMigrate** describes all of data store objects in the file. Alternatively, you can provide a list of names of data store objects on the command line. The names in this list must be fully qualified and cannot use wildcard characters.

TimesTen to Oracle data type conversions

Both TimesTen and Oracle data types are supported in TimesTen 7.0 When migrating a data store from an earlier version of TimesTen to TimesTen release 7.0, you can convert the data types in your data store to the default Oracle type mode. This is not required, however.

In replication, the type mode must be the same on both sides of the replication scheme. Therefore you cannot convert the data types as part of an online upgrade, as TimesTen releases previous to 7.0 do not support Oracle data types.

Note: If -convertTypesToOra is specified, and a DECIMAL (or NUMERIC) column exists in the data store with a precision > 38, the column is converted to a NUMBER column with a precision of 38, and a warning is returned. If this occurs, and column values exist that will overflow or underflow with a precision of 38, those values are reduced or increased to the maximum or minimum possible value for a NUMBER with a precision of 38. Because of this and some other cases, the data type conversion procedures (using -convertTypesToOra and -convertTypesToTT) are not guaranteed to be reversible. Converting types from TT->ORA->TT can result in columns and data which are different from the original in some cases.

To convert from TimesTen data types to Oracle data types, use the -convertTypesToOra option.

The `-convertTypesToOra` option instructs ttMigrate to make the following type conversions as it saves or restores tables:

From TimesTen Type	To Oracle Type
TT_CHAR	ORA_CHAR
TT_VARCHAR	ORA_VARCHAR2
TT_NCHAR	ORA_NCHAR
TT_NVARCHAR	ORA_NVARCHAR2
TT_DECIMAL	ORA_NUMBER
TT_DATE	ORA_DATE (append 12:00:00 am)
TT_TIMESTAMP	ORA_TIMESTAMP(6)

Note: Columns of type TT_TINYINT, TT_SMALLINT, TT_INTEGER, TT_BIGINT, BINARY_FLOAT, BINARY_DECIMAL, TT_BINARY, TT_VARBINARY, and TT_TIME are not converted.

For information on data types, see Chapter 1, “Data Types” in the *Oracle TimesTen In-Memory Database SQL Reference Guide*,

Oracle to TimesTen data type conversions

When migrating tables backward from TimesTen release 7.0 to an earlier version of TimesTen, you may need to convert Oracle data types to TimesTen data types, as the Oracle data types were not supported in releases prior to 7.0

To convert from Oracle data types to TimesTen data types, use the -convertTypesToTToption.

The `-convertTypesToTT` option instructs the ttMigrate utility to make the following type conversions as it saves or restores tables:

From Oracle Type	To TimesTen Type
ORA_CHAR	TT_CHAR
ORA_VARCHAR2	TT_VARCHAR
ORA_NCHAR	TT_NCHAR
ORA_NVARCHAR2	TT_NVARCHAR
ORA_NUMBER	TT_DECIMAL
ORA_DATE	TT_DATE (time portion of date is silently truncated)
ORA_TIMESTAMP	TT_TIMESTAMP

For information on data types, see Chapter 1, “Data Types” in the *Oracle TimesTen In-Memory Database SQL Reference Guide*.

Cache group data type conversions

When restoring a data store 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 TT Type	Oracle Type	Converted Type
TINYINT	NUMBER(p,s) when s > 0	NUMBER(p,s)
	Otherwise	TT_TINYINT

Pre-7.0 TT Type	Oracle Type	Converted Type
SMALLINT	NUMBER(p,s) when s > 0	NUMBER(p,s)
	Otherwise	TT_SMALLINT
INTEGER	NUMBER(p,s) when s > 0	NUMBER(p,s)
	Otherwise	TT_INTEGER
BIGINT	NUMBER(p,s) when s > 0	NUMBER(p,s)
	Otherwise	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[a]	BINARY_FLOAT
DOUBLE	Any[1]	BINARY_DOUBLE
FLOAT(x) x <=24	Any[1]	BINARY_FLOAT
FLOAT(x) x >= 24	Any[1]	BINARY_DOUBLE
CHAR(x)	Any[1]	ORA_CHAR(x)
VARCHAR(x)	Any[1]	ORAVARCHAR2(x)
BINARY(x)	Any[1]	TT_BINARY(x)
VARBINARY(x)	Any[1]	TT_VARBINARY(x)
DATE	DATE	ORA_DATE
TIMESTAMP	DATE	ORA_DATE
TIME	DATE	ORA_DATE
Any[1]	TIMESTAMP(m)	ORA_TIMESTAMP(m)

a. “Any” means the type value does not affect the converted result type.

For information on data types, see Chapter 1, “Data Types” in the *Oracle TimesTen In-Memory Database SQL Reference Guide* and "Differences between Oracle and TimesTen tables" in Chapter 8, “Compatibility Between TimesTen and Oracle” of the *TimesTen Cache Connect to Oracle Guide*.

Examples

The following command dumps all data store objects from data store `SalesDS` into a file called `sales.ttm`. If `sales.ttm` already exists, **ttMigrate** overwrites it.

```
ttMigrate -c DSN=SalesDS sales.ttm
```

This command appends all data store objects in the `SalesDS` data store owned by user `MARY` to `sales.ttm`:

```
ttMigrate -a DSN=SalesDS sales.ttm MARY.%
```

This command restores all data store objects from `sales.ttm` into the `SalesDS` data store:

```
ttMigrate -r DSN=SalesDS sales.ttm
```

This command restores `MARY.PENDING` and `MARY.COMPLETED` from `sales.ttm` into `SalesDS` (Case is ignored in migrate objects.):

```
ttMigrate -r DSN=SalesDS sales.ttm MARY.PENDING  MARY.COMPLETED
```

This command lists all migrate objects saved in `sales.ttm`:

```
ttMigrate -l sales.ttm
```

Notes

When migrating backwards 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.

When migrating data stores that contain cache groups from a previous release of TimesTen to TimesTen 7.0 or greater, you must use the option `-convertTypesToOra`. See Cache group data type conversions for a description of the data type mapping.

Character columns in cached tables must have not only the same length but also the same byte semantics as the underlying Oracle tables. Cache group migration fails when there is a mismatch in the length or length semantics of any of its cached tables.

Character sets: By default, ttMigrate stores table data in the database character set, unless you have specified the -saveAsCharset option. At restore time, conversion to another character set can be achieved by migrating the table into a data store 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 already 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 data stores 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 data store uses a character set whose maximum byte-length is 4 and the destination data store uses a character set whose maximum byte-length is 2.

TimesTen issues a warning whenever character set conversion takes place to alert you to the possibility of data loss due to conversion.

Data type conversions:

When migrating data from a pre-7.0 release of TimesTen, you must explicitly request data type conversions, using either the -convertTypesToOra or the -convertTypesToTT options.

ttMigrate saves the length semantic annotation (BYTE or CHAR) of CHAR and VARCHAR columns and restores these annotations when restoring into TimesTen releases that support them. When migrating backwards into a TimesTen release that does not support these annotations, columns with CHAR length semantics are converted to BYTE length, but their lengths are adjusted to match the byte length of the original columns. When migrating forward from a release that does not support these annotations, BYTE length semantics are used.

Foreign key dependencies: In restore mode, the presence of foreign key dependencies between tables may require **ttMigrate** to reorder tables to ensure that a child table is not restored before any of its parents. Such dependencies can also prevent a child table from being restored if any of its parent tables were not restored. For example, when restoring a table A that has a foreign key dependency on a table B, **ttMigrate** first checks to verify that table B exists in the data store. 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 t-trees as primary-key indexes into TimesTen releases that support this feature. When migrating backwards into a release that does not support t-trees 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 t-trees as primary-key indexes, the primary keys are restored as hash indexes of the same size as the original index.

INLINE columns: When migrating TimesTen tables that contain INLINE variable length columns to a release of TimesTen that is earlier than 5.1, you must explicitly use the `-noRepUpgrade` option. Using the default `-repUpgrade` option results in an error. The INLINE attribute of the columns is maintained, unless you specify otherwise using the `-inline` option.

Replication: Before attempting a full store migrate of replicated stores, make sure the host name and data store name are the same for both the source and destination data stores.

Other considerations: Because **ttMigrate** uses a binary format, you cannot use ttMigrate to:

- Migrate data stores between hardware platforms.
- Restore data saved with **ttBackup** or use **ttBackup** to restore data saved with **ttMigrate**.

Platforms: You can use **ttMigrate** to migrate data stores between 32- and 64-bit platforms if the two platforms are otherwise the same. Follow the examples in the *TimesTen Installation Guide*.

- On Windows, you can use **ttMigrate** to access data stores from any release of TimesTen. On Windows, this utility is supported for all TimesTen Data Manager and Client DSNs.

- On UNIX, the release of **ttMigrate** must match the release of the data store you are connecting to.

See Also ***ttBackup*** (page 83). The **ttBackup** utility backs up a data store. It is useful for creating a snapshot of a data store in a particular state.

ttBulkCp (page 86). The **ttBulkCp** utility saves a table to an ASCII file. It is useful for adding rows to a table, migrating TimesTen data stores between platforms and importing data from other applications.

ttRestore (page 168). The **ttRestore** utility restores a data store that has been backed up using the **ttBackup** utility.

ttmodinstall

Description Modifies the port number or enables Access Control for an instance of TimesTen. Useful if you discover that other process are listening on the port that you assigned to TimesTen at installation times.

Access Control If Access Control is enabled for your instance of TimesTen, this utility requires instance administrator privileges.

Syntax

```
ttmodinstall [-h | -help | -?]
ttmodinstall [-V | -version]
ttmodinstall -port portNumber
ttmodinstall -enableAccessControl
ttmodinstall -changeOracleSettings
```

Options **ttmodinstall** has the option:

`-changeOracleSettings`	Prompts the user to specify the value of ORACLE_HOME. This allows TimesTen to modify the specified location of the Oracle installation after installation.

`-h`
`-help`
`-?`
	Displays help information.

`-enableAccessControl`	Enables Access Control if it is not enabled in your TimesTen instance.
`-port` *`portNumber`*	Changes the daemon port for the current instance of TimesTen to *`portNumber`*.
`-V | -version`	Display TimesTen version information.

Examples To change the port number of the TimesTen instance to 12345, use:

```
ttmodinstall -port 12345
```

Note All TimesTen operations must be completely shut down to use this utility. This utility stops and then restarts the TimesTen daemon before making any changes to the instance.

Access Control cannot be disabled once TimesTen has been installed with Access Control enabled.

ttRepAdmin

Description

Displays existing replication definitions and monitors replication status. The **ttRepAdmin** utility is also used when upgrading to a new release of TimesTen, as described in the *Oracle TimesTen In-Memory Database Installation Guide*.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility requires ADMIN privileges or data store object ownership. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttRepAdmin [-h | -help | -?]

ttRepadmin [-V | -version]

ttRepAdmin -self -list [-scheme [owner.]schemeName]
      {DSN | -connStr connectionString}


ttRepAdmin -receiver [-name receiverName]
      [-host receiverHostName] [-state receiverState] [-reset]
      [-list] [-scheme [owner.]schemeName]
      {DSN | -connStr connectionString}


ttRepAdmin -log {DSN | -connStr connectionString}

ttRepAdmin -showstatus {DSN | -connStr connectionString}

ttRepAdmin -showconfig {DSN | -connStr connectionString}

ttRepAdmin -bookmark {DSN | -connStr connectionString}

ttRepAdmin -wait [-name receiverName] [-host receiverHostName]
      [-timeout seconds] {DSN | -connStr connectionString}


ttRepAdmin -duplicate -from srcDataStoreName
      -host srcDataStoreHost [-memcpy]
      [-setMasterRepStart] [-ramLoad]       [-delXla]
      [-UID userId] [-PWD pwd | -PWDCrypt encryptedPwd]
      [-drop { [owner.]table … | [owner.]sequence |ALL }]
      [-truncate { [owner.]table … | ALL }]
      [-compression 0 | 1] [-bandwidthmax maxKbytesPerSec]
      [-keepCG [-cacheUid cacheUid [-cachePwd cachePwd]]
         [-recoveringNode | -deferCacheUpdate]
      | -nokeepCG]
      [-remoteDaemonPort portNo] [-verbosity {0|1|2}]
      [-localhost localHostName]
      {destDSN | -connStr connectionString}
```

ttRepAdmin Operations

The **ttRepAdmin** utility is the “swiss army knife” replication utility that can be used for a number of replication operations. These operations fall into the following categories:

- Help and version information
- Data store information
- Subscriber data store operations
- Duplicate a data store
- 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]
```

`-h`
`-help`
`-?`
	Display help information.

`-V | -version`	Display TimesTen version information.

Data store information

Use this form of **ttRepAdmin** to obtain summary information about a data store.

```
ttRepAdmin -self -list [-scheme [owner.]schemeName]
{DSN | -connStr connectionString}
```

Options **ttRepAdmin** -self -list has the options:

DSN	Data source name of a master or subscriber data store.
-connStr *connectionString*	Connection string of a master or subscriber data store.
-self	Specified data store.
-list	Lists data store 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 data store `my_dsn`.

Subscriber data store operations

Use this form of **ttRepAdmin** to check the status or reset the state of a subscriber (receiver) data store.

```
ttRepAdmin -receiver [-name receiverName]
       [-host receiverHostName]
       [-state receiverState] [-reset]
       [-list] [-scheme [owner.]schemeName]
       {DSN | -connStr connectionString}
```

Options **ttRepAdmin** -receiver has the options:

DSN	Data source name of the master data store.
-connStr *connectionString*	Connection string of the master data store.
-receiver	Subscriber data stores receiving updates from the master. Use -name and -host to specify a specific subscriber data store.
-name *receiverName*	A specific subscriber (receiving) data store. The *receiverName* is the last component in the data store path name.
-host *receiverHostName*	Hostname or TCP/IP address of the subscriber host machine.
-state start -state stop -state pause	Sets the state of replication for the subscriber. • start— Starts replication to the subscriber (default). • stop— Stops replication to the subscriber, discarding updates. • pause— Pauses the replication agent, preserving updates. See the section, "Setting the replication state of subscribers"in the *TimesTen to TimesTen Replication Guide* for more information.
-reset	Clears the bookmark in the master data store 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 data store is changed, such as when the data store is re-created using **ttMigrate** or **ttBulkCp**. If the master data store 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 of the subscribers of the master data store, my_dsn.

```
ttRepAdmin -receiver -name rep_dsn -list my_dsn
```

The above syntax lists replication information for the `rep_dsn` subscriber of the master data store, 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 data store. Should only be used when migrating a replicated data store 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 data store to the Start state with respect to the master data store, `my_dsn`.

Duplicate a data store

Use this form of **ttRepAdmin** to create a new data store with the same contents as the master data store.

Before running the `ttRepAdmin -duplicate` command, use **ttStatus** to be sure that the replication agent is started for the source data store.

```
ttRepAdmin -duplicate -from srcDataStoreName
      -host srcDataStoreHost [-memcpy]
      [-setMasterRepStart] [-ramLoad]       [-delXla]
      [-UID userId] [-PWD pwd | -PWDCrypt encryptedPwd]
      [-drop { [owner.]table … | [owner.]sequence |ALL }]
      [-truncate { [owner.]table … | ALL }]
      [-compression 0 | 1] [-bandwidthmax maxKbytesPerSec]
      [-keepCG [-cacheUid cacheUid [-cachePwd cachePwd]]
         [-recoveringNode | -deferCacheUpdate]
      | -nokeepCG]
      [-remoteDaemonPort portNo] [-verbosity {0|1|2}]
      [-localhost localHostName]
      {destDSN | -connStr connectionString}
```

Options **ttRepAdmin** -duplicate has the options:

`-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.
`-compression 0 | 1`	Enables or disables compression during the duplicate operation. The default is 0 (disabled).
`-connStr` *`connectionString`*	Specifies the connection string of the destination data store.
`-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 data store.
`destDSN`	Indicates the data source name of the destination data store.
`-drop { [`*`owner`*`.]table … | [`*`owner`*`.]sequence | ALL`	Drops any tables or sequences that are copied as part of the -duplicate operation but which are not included in the replication scheme. The option is ignored if the table is a cache group table.
`-duplicate`	Creates a duplicate of the specified data store using replication to transmit the data store contents across the network. See the sections, "Copying a master data store to a subscriber" and "Recovering a failed data store" in the *TimesTen to TimesTen Replication Guide* for more information. When using this option, TimesTen tables in a cache group are duplicated as regular TimesTen tables in the new data store unless you use the `-keepCG` option.
`-from` *`srcDataStoreName`*	Used with `-duplicate` to specify the name of the sender (or master) data store. The *`srcDataStoreName`* is the last component in the data store path name.
`-host` *`srcDataStoreHost`*	Defines the hostname or TCP/IP address of the sender (or master) data store.

`-keepCG [-cacheUid` *`cacheUid`* `-cachePwd` *`cachePwd`*`] [-recoveringNode | -deferCacheUpdate] | -noKeepCG`	Specifies whether tables in the specified cache group should be maintained as cache group tables or converted to regular tables. The *`cacheUid`* is the ID for the cache administration user and the *`cachePwd`* is the associated password. If no password is provided, **ttRepAdmin** prompts for a password. The default is `-noKeepCG`. `-recoveringNode` specifies that `-duplicate` is being used to recover a failed node for a replication scheme that includes an AWT or autorefresh cache group. Do not specify `-recoveringNode` when rolling out a new or modified replication scheme to a node. If `-duplicate` cannot update metadata stored on the Oracle database and all incremental autorefresh cache groups are replicated, then updates to the metadata will be automatically deferred until the cache and replication agents are started. `-deferCacheUpdate` forces the deferral of changes to metadata stored on the Oracle database until the cache and replication agents are started and the agents can connect to the Oracle database. Using this option can cause a full autorefresh if some of the incremental cache groups are not replicated or if `-duplicate` is being used for rolling out a new or modified replication scheme to a node.
`-localhost` *`hostName`*	Can be used with `-duplicate` and `-setMasterRepStart` to explicitly identify the name or IP address of the local host.
`-memcpy`	To be used with `-duplicate` when duplicating a diskless data store to reduce the time during which the remote data store is locked.
`-PWD` *`pwd`*	If Access Control is enabled, the password of the user specified in the UID option.
`-PWDCrypt` *`encryptedPwd`*	If Access Control is enabled, the encrypted password of the user specified in the UID option.

`-ramLoad`	Keeps the data store in memory upon completion of the duplicate operation. This option avoids the unload/reload data store cycle to improve the performance of the duplicate operation when copying large data stores. After the duplicate option, the RAM Policy for the data store is set to "manual." Use the **ttAdmin** utility to make further changes to the RAM policy.
`-remoteDaemonPort` *`portNo`*	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 data stores that have stores which use automatic port configuration.
`-setMasterRepStart`	When used with `-duplicate`, this option sets the replication state for the newly created data store to the Start state just before the data store is copied across the network. This ensures that all updates made to the source data store after the duplicate operation are replicated to the newly duplicated local data store. Any unnecessary log files for the data store are removed.
`-truncate` `[`*`owner`*`.]table … | ALL`	Truncates any tables that are copied as part of the `-duplicate` operation but which are not included in the replication scheme. The option is ignored if the table is a cache group table.
`-UID` *`userid`*	If Access Control is enabled, the user ID of a user having the ADMIN privilege on the peer to be copied must be supplied.
`-verbosity {0 | 1 | 2}`	Provide details of the communication steps within the duplicate process and reports progress information about the duplicate transfer. **0** - No diagnostics are returned (default) **1** - Reports details of the duplicate parameters to `stdout`. **2** - Reports details of the duplicate parameters and details of the duplicate transfer operation to `stdout`.

Examples:

```
ttRepAdmin -duplicate -from srcDataStoreName -host srcDataStoreHost
      [destDSN | -connStr connectionString]
```

The above syntax creates a replica of a remote replication-enabled temporary or permanent data store. This operation may be used to automate the creation of a backup data store.

If the remote data store is diskless, this operation obtains a data store-level lock for the entire period of the memory transfer on the remote source data store. It prevents any other connection from accessing the remote data store during the duplicate operation.

In the above syntax of **ttRepAdmin**,

- *destDSN* is the local DSN.
- *connectionString* is the connection string for the local DSN.
- *srcDataStoreName* is the remote source data store name. This name is the last component of the data store path name.
- *srcDataStoreHost* is the TCP/IP host name of the machine where remote source data store is located.

```
ttRepAdmin -duplicate -from srcDataStoreName
      -host srcDataStoreHost -memcpy
      [destDSN | -connStr connectionString]
```

In the case where the remote data store is diskless, you may use the `-memcpy` option as shown above to create a replica of a remote replication-enabled temporary or permanent data store. With this additional option, ttRepAdmin `-duplicate` in diskless mode makes an intermediate copy of the remote data store to memory prior to copying the data store across the network. The remote data store is locked only for the duration of the memory-to-memory copy. This reduces the database lock out time for this operation in diskless mode. In cases where the memory-to-memory bandwidth on the remote machine exceeds the network data bandwidth between the remote machine and the local machine, the locking time on the remote machine is reduced. An additional requirement is that there be adequate RAM available for the remote extra copy.

```
ttRepAdmin -duplicate -from srcDataStoreName
      -host srcDataStoreHost
      -setMasterRepStart -localhost localHostName -ramLoad
      [destDSN | -connStr connectionString]
```

In the above syntax, the `-setMasterRepStart` option causes the replication state in the *srcDataStoreName* data store to be set to the Start state before it is copied across the network, and then keeps the data store in memory. You can use the `-localhost` option to explicitly identify the local host, which is useful if the local host uses a non-standard name, such as an IP address. These options ensure that all updates made after the duplicate operation are replicated from the remote data store to the newly created or restored local data store.

In the above syntax of **ttRepAdmin**,

- *localHostName* is the hostname or TCP/IP address of the local machine.

Note This utility can duplicate any temporary table definition in a data store, but not its instances.

Replication status

Use this form of **ttRepAdmin** to check the size of the log files, bookmark position, or replication configuration of a master data store.

```
ttRepAdmin -log {DSN | -connStr connectionString}
ttRepAdmin -showstatus {DSN | -connStr connectionString}
ttRepAdmin -showconfig {DSN | -connStr connectionString}
ttRepAdmin -bookmark {DSN | -connStr connectionString}
```

Options **The ttRepAdmin** monitor operations have the options:

DSN	Indicates the data source name of the master data store.
`-connStr` *`connectionString`*	Specifies the connection string of the master data store.
`-log`	Prints out number and size of log files retained by replication to transmit updates to other data stores.
`-showconfig`	Lists the entire replication configuration. See the section, "Show configuration of replicated data stores" in the *TimesTen to TimesTen Replication Guide* for more information.
`-showstatus`	Reports the current status of the specified replicated data store. See the section, "Show replication status" in the *TimesTen to TimesTen Replication Guide* for more information.
`-bookmark`	Reports the earliest log sequence number that replication needs to read, the most recently created log sequence number, and the latest log sequence number whose record has been flushed to disk. See the section, "Show replicated log records" in the *TimesTen to TimesTen Replication Guide* for more information.

Examples: `ttRepAdmin -log DSN`

The above syntax reports the number of log files that replication is retaining to transmit updates to other data stores. The replication agent retains a log file until all updates in that log file have been successfully transferred to each subscriber data store.

```
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 data store for the specified DSN. The output includes the state of all of the threads in the replication agents for the replicated data stores, bookmark locations, port numbers, and communication protocols.

```
ttRepAdmin -bookmark DSN
```

The above syntax prints out the log sequence numbers of the earliest log record still needed by replication, the last log record written to disk, and the last log record generated.

Wait for updates to complete

Use this form of **ttRepAdmin** to assure that all of the updates in the log are replicated to all subscribers before call returns.

```
ttRepAdmin -wait [-name receiverName] [-host receiverHostName]
       [-timeout seconds] {DSN | -connStr connectionString}
```

Options **ttRepAdmin** -wait has the options:

DSN	Indicates the data source name of the master data store.
-connStr *connectionString*	Specifies the connection string of the master data store.
-wait	Waits for replication to become current before continuing.
-name *receiverName*	Identifies the data store. The data store name is the last component in the data store path name.
-host *receiverHostName*	Defines the hostname or TCP/IP address of the subscriber host machine.
-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 data store. 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 acknowledgement 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 data store.

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 data store.

Using SQL configuration, you can create multiple replication schemes in the same data store. 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 data stores with the same name on different hosts, you can indicate which data store you wish to operate on by using `-host`. For example, if all the subscribers have the name DATA, you can set the replication state on host SW1 with:

```
ttRepAdmin -receiver -name DATA -host SW1 -state start DSN
```

See Also

For a full description of TimesTen Replication, see the *TimesTen to TimesTen Replication Guide*.

For upgrade examples, see Chapter 3, “ in the *Oracle TimesTen In-Memory Database Installation Guide*.

ttRestore

Description

Creates a data store from a backup that has been created using the **ttBackup** utility. If the data store already exists, **ttRestore** does not overwrite it.

The data store attributes in the **ttRestore** connection string can contain any of the First Connection or General Connection attributes. It can also include the Data Store attribute: **LogDir**. All other data store attributes are copied from the backup files. This allows the restored data store to be relocated.

The **ttRestore** action is somewhat more powerful than a first connect, as it can move the data store. It is somewhat less powerful than creating a new data store, as it cannot override the other Data Store attributes.

For an overview of the TimesTen backup and restore facility, see "Copying, migrating, backing up and restoring a data store" in the *Oracle TimesTen In-Memory Database Operations Guide*.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility requires ADMIN privileges or data store object ownership. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttRestore [-h | -help | -?]
ttRestore [-V | -version]
ttRestore [-fname filePrefix] [-noconn] -dir directory
{DSN | -connStr connectionString}
ttRestore -i [-noconn] {DSN | -connStr connectionString}
```

Options

ttRestore has the options:

`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Specifies an ODBC data source name of the data store to be administered.
`-dir` *`directory`*	Specifies the directory where the backup files are stored.
`-fname` *`filePrefix`*	Specifies the file prefix for the backup files in the backup directory. The backup files must have been stored in the backup directory with this prefix. The default value for this parameter is the file name portion of the `DataStore` parameter of the data store's ODBC definition.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-i`	Read standard input for the backup data. You cannot use the `-dir` or `-fname` options with `-i`. **ttRestore** rolls the logs forward.
`-noconn`	In order to ensure that the restore was successful, **ttRestore** connects to the data store as a last step. This option disables that last connect.
`-V | -version`	Prints the release number of **ttRestore** and exits.

Example

```
ttRestore -dir /users/pat/TimesTen/backups
-fname FastInsBkup “DSN=FastIns”
```

To backup a data store named `origDSN` to the directory `/users/rob/tmp` and restore it to data store named `restoredDSN`, use:

```
ttBackup -dir /users/rob/tmp -fname restored "dsn=origDSN"
ttRestore -dir /users/rob/tmp -fname restored "dsn=restoredDSN"
```

The value of `fname` is the name that you want for the prefix portion of the backup file name.

On Unix, to restore a tape backup to the `FastIns` data store, use:

```
dd bs=64k if=/dev/rmt0 | ttRestore -i DSN=FastIns
```

Note

This utility is supported only where the TimesTen Data Manager is installed.

See Also

“ttBackup” on page 83
“ttBulkCp” on page 86
“ttMigrate” on page 136

ttSchema

Description

Prints out the schema, or selected objects, of a data store. 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

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 data store rather than a history of how the data store came to arrive at its current state, perhaps through ALTER statements. An entire data store, including data, cannot be completely reconstructed from the output of **ttSchema**. The output of **ttSchema** can be played back by the **ttIsql** utility in order to rebuild the full schema of a data store.

Syntax

```
ttSchema [-h | -help | -?]
ttSchema [-V | -version]
ttSchema [-l] [-c] [ -list {all | tables | views | sequences |
         cachegroups | repschemas } [,...] ]
         [-st | -systemTables] [-u TT_instance]
         [-connStr connection_string | DSN ]
         [[owner.]name][...]
```

Options

ttSchema has the options:

–connStr *connection_string*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
-c	Compatibility mode. Limits the use of TimesTen-specific and release-specific keywords and extensions. This may be useful if the **ttSchema** output is being used as input to an older TimesTen release, or to some other database system, such as Oracle. For this release, -c prevents the INLINE and NOT INLINE keywords from being output.
DSN	Specifies an ODBC data source name of the data store from which to get a schema.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`–l`	One per-line listing of objects in the data store.
`-list {all | tables | views | sequences | cachegroups | repschemas }[,...]`	A comma-separated list of objects to generate. Lists only those types of objects specified. Default is `-list all`.
`[owner.]`*`name`*	Limits the scope of the output to specified data store object(s).
`-st | -systemTables`	Include system tables. By default, they are omitted.
`-u` *`TT_instance`*	Outputs the user definitions and privilege information in the form of SQL statements that can be used to recreate the user environment within a different instance. The -u option emits output when run against a data store that is Access Control enabled. The actual passwords are not included in this information. TimesTen substitutes an empty password.
`-V | -version`	Prints the release number of **ttSchema** and exits.

Examples Suppose a datastore called ORDERS is created with the SQL statements:

```
CREATE TABLE xyz.customer (
  cust_num          INTEGER NOT NULL PRIMARY KEY,
  region            CHAR(2) NOT NULL,
  name              VARCHAR(80),
  address           VARCHAR(255) NOT NULL
)
UNIQUE HASH ON (cust_num) PAGES=100;

CREATE SEQUENCE xyz.custID MINVALUE 1 MAXVALUE 1000000;

CREATE TABLE xyz.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 xyz.customer (cust_num)
)
UNIQUE HASH ON (ord_num) PAGES=100;

CREATE MATERIALIZED VIEW xyz.order_summary AS
  SELECT cust.name, ord.ord_num, count(*) ord_count
  FROM xyz.orders ord, xyz.customer cust
  WHERE ord.cust_num = cust.cust_num
```

```
    GROUP BY cust.name, ord.ord_num;
CREATE REPLICATION xyz.rep
  ELEMENT e1 TABLE xyz.orders MASTER rep1 ON "box1"
    SUBSCRIBER rep2 ON "box2"
  ELEMENT e2 TABLE xyz.orders MASTER rep2 ON "box2"
    SUBSCRIBER rep1 ON "box1";
```

The output of ttSchema would be:

```
% ttSchema ORDERS
CREATE TABLE xyz.customer (
  cust_name INTEGER NOT NULL,
  region CHAR(2) NOT NULL,
  “name” VARCHAR(80),
  address VARCHAR(255) NOT INLINE NOT NULL,
  PRIMARY KEY(cust_num))
  UNIQUE HASH ON(cust_num) PAGES = 100;
CREATE TABLE xyz.orders (
  ord_num INTEGER NOT NULL,
  cust_num INTEGER NOT NULL,
  when_placed TIMESTAMP NOT NULL,
  when_shipped TIMESTAMP,
  PRIMARY KEY(ord_num),
  FOREIGN KEY(cust_num) REFERENCES xyz.customer (cust_num))
  UNIQUE HASH ON (ord_num) PAGES = 100;
CREATE SEQUENCE xyz.custid
  INCREMENT BY 1
  MINVALUE 1
  MAXVALUE 1000000
  START 1;
CREATE REPLICATION xyz.rep
  ELEMENT e1 TABLE xyz.orders
    MASTER rep1 on "box1"
    SUBSCRIBER rep2 ON "box2"
  ELEMENT e2 TABLE xyz.orders
    MASTER rep2 ON "box2"
    SUBSCRIBER rep1 ON "box1"
;
CREATE MATERIALIZED VIEW xyz.order_summary AS
  SELECT cust."name" name, ord.ord_num ord_num,
    COUNT(*) ord_count FROM xyz.orders ord, xyz.customer cust
    WHERE ord.cust_num = cust.cust_num
    GROUP BY cust."name", ord.ord_num;
```

The command:

```
% ttSchema -list sequences ORDERS
```

limits the output to

```
CREATE SEQUENCE xyz.custid
  INCREMENT BY 1
  MINVALUE 1
  MAXVALUE 1000000
  START 1;
```

To list the schema by object name,

```
% ttSchema ORDERS xyz.rep
CREATE REPLICATION xyz.rep
  ELEMENT e1 TABLE xyz.orders
    MASTER rep1 ON "box1"
    SUBSCRIBER rep2 ON "box2"
  ELEMENT e2 TABLE xyz.orders
    MASTER rep2 ON "box2"
    SUBSCRIBER rep1 ON "box1";
```

Notes

The SQL generated does not produce a history of transformations through ALTER statements, nor does it preserve table partitions, although the output gives information on table partitions in the form of SQL comments.

Output is not guaranteed to be compatible with DDL recognized by previous releases of TimesTen.

ttSize

Description

Estimates the amount of space that a given table, including any views in the data store will consume when it grows to include *rows* rows. It can be used on existing tables or to estimate table sizes when creating tables. If no owner is specified, the size information is printed for all tables of the given name. The size information includes space occupied by any indexes defined on the table.

The memory required for varying-length columns is estimated by using the average length of the columns in the current table as the average length of the columns in the final table. If there are no rows in the current table, then **ttSize** assumes that the average column length is one half the maximum column length.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility requires SELECT privileges or data store object ownership. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttSize [-h | -help | -?]
ttSize [-V | -version]
ttSize -tbl [owner.]tableName [-rows rows]
{-connStr connection_string | DSN}
```

Options

ttSize has the options:

–connStr *connection_string*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
DSN	Specifies the name of a data source to which **ttSize** should connect to retrieve table information.
-h -help -?	Prints a usage message and exits.
–rows *rows*	Specifies the expected number of rows in the table. Space required to store a TimesTen table includes space for the actual data, plus overhead for bookkeeping, dynamic memory allocation and indexes. TimesTen may consume additional space due to memory fragmentation, temporary space allocated during query execution and space to hold compiled SQL statements. If this option is omitted, the number of rows in the existing table is used, or 1 if the table is empty.

`–tbl [owner.]tableName`	Specifies the name of the table whose definition should be used for size estimation. If the owner is omitted, the login name of the user is tried. If that is not found, the user SYS is used.
`-V | -version`	Prints the release number of **ttSize** and exits.

Example

To estimate the space required for a table, create the table in TimesTen, populate it with a sample of representative rows, create desired indexes and execute **ttSize** with those definitions. For example, to estimate the size of the `NAMEID` table in the data source `FixedDs` when it grows to 200,000 rows, execute:

```
ttSize FixedDs -tbl Nameid -rows 200000
```

Notes

Another method for estimating size requirements and measuring fragmentation is to use the MONITOR table. (See "SYS.MONITOR" in *Oracle TimesTen In-Memory Database SQL Reference Guide*.)

The columns PERM_ALLOCATED_SIZE and PERM_IN_USE_SIZE show the currently allocated size of the data store (in KB units) and the in-use size of the data store. The system updates this information each time a connection is made or released and each time a transaction is committed or rolled back.

This utility is supported only for TimesTen Data Manager DSNs. It is not supported for TimesTen Client DSNs.

See also

"ttSize" on page 324

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 data stores.
- Number of users currently connected to each TimesTen data store.
- The RAM and replication policies.
- TimesTen cache agent status.
- TimesTen Webserver address.
- Miscellaneous status information.

Syntax

```
ttStatus [-h | -help | -?]
ttStatus [-V | -version]
ttStatus [-v] [-r secs] [-[no]pretty]
```

Options

ttStatus has the options:

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-[no]pretty`	Do [not] use pretty formatting. The default is pretty formatting, which uses the values of the **ConnectionName** attribute.
`-r secs`	Allows **ttStatus** to continue running. Updates status report every *secs* seconds.
`-V | -version`	Prints the release number of **ttStatus** and exits.
`-v`	Prints detailed information that is useful for TimesTen customer support.

Sample Output

When you invoke the command, a report that describes the current state of the system is displayed to `stdout`. The following is sample output:

```
TimesTen status report as of Wed Aug 13 16:47:56 2003

Daemon pid 15808 port 15000 instance -
TimesTen server pid 16888 started on port 15102
------------------------------------------------------------
Data store /tmp/rep1
There are 2 connections to the data store
Data store is in shared mode
```

```
Shared Memory KEY 0x060036f0 ID 152910
Process pid 19180 context 0xacd7e8 connected (KEY 0x060036f0)
Subdaemon pid 15035 context 0x62328 connected (KEY 0x0c000fdf)
Subdaemon pid 15035 context 0xdadb8 connected (KEY 0x0c000fdf)
Subdaemon pid 15035 context 0xfadc8 connected (KEY 0x0c000fdf)
Replication policy: Manual
Cache agent restart policy: manual
------------------------------------------------------------
Data store /tmp/temp51
There are 2 connections to the data store
Data store is in shared mode
Shared Memory KEY 0x040036f0 ID 152810
Subdaemon pid 16604 context 0x3ab60 connected (KEY 0x040036f0)
Subdaemon pid 16604 context 0xdadb8 connected (KEY 0x040036f0)
Subdaemon pid 16604 context 0xfadc8 connected (KEY 0x040036f0)
Server pid 19376 context 0x48f0068 connected (KEY 0x040036f0)
(Client Information: pid 19776; IPC: SHMIPC;
         Node: xyz (10.10.12.192))
Replication policy: Manual
Cache agent restart policy: manual
------------------------------------------------------------
Data store /tmp/rep2
There are no connections to the data store
Replication policy: Manual
Cache agent restart policy: manual
------------------------------------------------------------
End of report
```

When you invoke the command with the -pretty option, a report that describes the current state of the system is displayed to `stdout`. The following is sample output:

```
Data store /ds0/kirke/sample
There are 5 connections to the data store
Data store is in shared mode
Shared Memory KEY 0x2c00c42d ID 37613
Type            PID     Context             Connection Name      ConnID
Process         29508   0x00000001001c6680  myconnection              1
Subdaemon       29505   0x0000000100165290  Worker                 2044
Subdaemon       29505   0x00000001001df190  Flusher                2045
Subdaemon       29505   0x0000000100206730  Checkpoint             2047
Subdaemon       29505   0x0000000010021cc50  Monitor                2046
Replication policy  : Manual
Cache agent policy : Manual
```

Notes While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

See also “ttAdmin” on page 75

ttSyslogCheck (UNIX)

Description

Determines if the system's `/etc/syslog.conf` file is properly configured for TimesTen. The TimesTen Data Manager uses `syslog` to log a variety of progress messages. It is highly desirable to configure `syslog` so that all TimesTen messages are written to disk in a single disk file. The **ttSyslogCheck** utility examines the `syslog` configuration (in `/etc/syslog.conf`) to verify that it is properly configured for TimesTen.

If `syslog` is properly configured, **ttSyslogCheck** displays the name of the file that TimesTen messages are logged to and exits with exit code 0. If `syslog` is not properly configured, **ttSyslogCheck** displays an error message and exits with code 1.

Syntax

```
ttSyslogCheck [-h | -help | -?]
ttSyslogCheck [-V | -version]
ttSyslogCheck [-facility name]
```

Options

ttSyslogCheck has the options:

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-facility` *`name`*	Specifies the `syslog` facility *`name`* being used for daemon logging.
`-V | -version`	Prints the release number of **ttSyslogCheck** and exits.

Note

This utility is supported only where the TimesTen Data Manager is installed.

ttTail

Description Fetches TimesTen internal trace information from a data store and displays it to `stdout`. By default, TimesTen generates no tracing information; see "ttTraceMon" on page 181 for more information.

Access Control If Access Control is enabled for your instance of TimesTen, this utility requires SELECT privileges or data store object ownership. If authentication information is not supplied in the connection string or DSN, this utility prompts for a user ID and password before continuing.

Syntax

```
ttTail [-h | -help | -?]
ttTail [-V | -version]
ttTail [-f] {-connStr connection_string | DSN}
```

Options The **ttTail** utility supports the options:

`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Indicates the ODBC data source name of the data store 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 data store's trace buffer to retrieve and display additional TimesTen trace records. For example, this is useful for generating a display of trace data that is updated in real time.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-V | -version`	Prints the release number of **ttTail** and exits.

Example

```
ttTail MyDatastore
```

Notes While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

ttTraceMon

Description

The **ttTraceMon** utility lets you enable and disable the TimesTen internal tracing facilities.

Tracing options can be enabled and disabled on a per data store basis. Each data store 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 a number of subcommands to enable, disable, dump and manipulate trace information. **ttTraceMon** can be executed either interactively—multiple subcommands can be entered at a prompt—or non 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 semi-colons. 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, listing no components means that the command operates on all components.

Syntax

```
ttTraceMon [-h | -help | -?]
ttTraceMon [-V | -version]
ttTraceMon [-e subcommand]
{-connStr connection_string | DSN}
```

Options

ttTraceMon has the options:

`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`DSN`	Indicates the ODBC data source name of the data store from which to get trace information.

`-e subcommand`	Causes the subcommand to be executed against the specified data store. If the subcommand consists of more than one word, enclose it in double quotes. For example: `ttTraceMon -e "show err" SalesData` Once the subcommand is complete, **ttTraceMon** exits. If `-e` is not specified, **ttTraceMon** starts in interactive mode, reading commands from `stdin` and displaying results to `stdout`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-V | -version`	Prints the release number of **ttTraceMon** and exits.

Sub-commands

ttTraceMon can be called with the following subcommands:

Command	Description
`components`	List the names and internal identifiers of all `components`.
`connection {all | self | connectionNum} [on |off]`	Turn tracing on/off for specified connection. At data store 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 data store object ownership.
`dump` *`comp`*	Prints all trace records for component *`comp`*. Requires SELECT privileges or data store 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 data store object ownership.

Command	Description
`outfile`	Prints the current outfile setting.
`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**.
`show`	Shows all the trace levels in force.
`show` *`comp`*	Shows the trace level for component *`comp`*.

Notes

Because tracing can degrade performance significantly, we recommend that you enable tracing only to debug problems. While primarily intended for use by TimesTen customer support, this information may be useful to system administrators and developers.

This utility is supported only where the TimesTen Data Manager is installed.

ttuser

Description

Prompts for a password and returns an encrypted password. You can then include the output in a connection string or as the value for the **PWDCrypt** attribute in an ODBCINI file.

Access Control

If Access Control is enabled for your instance of TimesTen, this utility does not require any particular privileges.

Syntax

```
ttuser [-h | -help | -?]
ttuser [-V | -version]
ttuser -pwdcrypt
```

Options

The **ttuser** utility supports the options:

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-pwdcrypt`	Generates an encrypted password value for the PWDCrypt attribute.
`-V | -version`	Prints the release number of **ttuser** and exits.

ttVersion

Description

The **ttVersion** utility lists the TimesTen release information, including: number, platform, instance name, instance admin, instance home directory, daemon home directory, port number and build timestamp.

Syntax

```
ttVersion
```

Options

ttVersion has the option:

`-m`	Generates machine-readable enhanced output. If not specified, abbreviated information is output.

Output

ttVersion produces the following sample output.

```
TimesTen Release 7.0 (32 bit Linux/x86) (terry:15100)
Sept 12 2006 09:20:46
```

ttVersion -m produces the following sample output. Most of the entries only appear for patch releases.

```
patched=yes
product=TimesTen
major=7
minor=0
patch=0
reldot4=0
reldot5=0
version=7.0.0.0.0
shortversion=70
numversion=070000000000
bits=32
os=Linux/x86
buildtstamp=1152549772
buildtime=2006-07-10T16:42:52Z
clientonly=no
instance=terry
effective_port=4146
orig_port=4146
instance_admin=terry
effective_insthome=
       /spider/daspinwa/ttcur/TTBuild/linux86_dbg/install
orig_insthome=/spider/terry/ttcur/TTBuild/linux86_dbg/install
effective_daemonhome=
       /spider/terry/ttcur/TTBuild/linux86_dbg/install/info
orig_daemonhome=
       /spider/terry/ttcur/TTBuild/linux86_dbg/install/info
access_control=0
```

ttXactAdmin

Description

The **ttXactAdmin** utility lists ownership, status, log and lock information for each outstanding transaction. The **ttXactAdmin** utility also allows you to heuristically commit, abort or forget an XA transaction branch.

Syntax

```
ttXactAdmin  [-h | -help | -?]

ttXactAdmin [-V | -version]

ttXactAdmin [-v verbosity] [-lsn]
           [-mt maxTrans] [-ml maxLocks] [-pid pid]
           [-xact xid] [-tbl [owner.]tableName]
           [-row rowid] [-interval seconds] [-count iterations]
           {DSN | -connstr <connStr>}

ttXactAdmin -latch
           [-interval <seconds>] [-count <number>]
           {DSN | -connstr connStr}

ttXactAdmin -latchstats [clear | off |on | show]
           [-interval seconds] [-count iterations]
           {DSN | -connstr connStr}

ttXactAdmin -connections

           [-pid pid]
           [-interval seconds] [-count iterations]

           {DSN | -connstr connStr}

ttXactAdmin -xactIdRollback xid {DSN | -connstr connStr}

ttXactAdmin {-HCommit xid | -HAbort xid | -HForget xid}
           {DSN | -connstr connStr}
```

Options

ttXactAdmin has the options:

`-connections`	Shows all current connections to the data store. When run with the `-connections` option, **ttXactAdmin** itself does not establish a true connection to the data store, and requires no latches. This can be useful when diagnosing frozen systems.
`-connStr` *`connection_string`*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
`-count` *`iterations`*	Generate the report *`iterations`* times. If no `-interval` option is specified, an interval of 1 second is used.

DSN	Indicates the ODBC data source name of the data store to be administered.
-h -help -?	Prints a usage message and exits.
-HAbort *xid*	Heuristically abort an XA transaction branch in TimesTen. The specified transaction ID must be the local TimesTen TransID.
-HCommit *xid*	Heuristically commit an XA transaction branch in TimesTen. The specified transaction ID must be the local TimesTen TransID
-HForget *xid*	Heuristically forget an XA transaction branch in TimesTen. The specified transaction ID must be the local TimesTen TransID.
-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 data store specified.
-latchstats [clear | off | on | show]	This option is to be used by TimesTen Customer Support only. Performs the requested latchstat operation. All other options are ignored when -latchstats is used. clear - Resets all latchstat information to zero. off - Turns off collection of latchstats. on - Turns on collection of latchstats. show - Shows the latch information, including access counts and other stats. Default, if no operation is specified.
-lsn	Displays the log sequence numbers of the first and last log records, if any, written by the transaction. If both LSN values are -1, it implies that the transaction is read-only. Requires SELECT privileges or data store object ownership.

`-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 p`*`id`*	Displays only transactions started by the process with the specified pid. On Linux, it is the pid of the thread that opens the connection.
`-row` *`rowid`*	Displays lock information for the specified row. Requires SELECT privileges or data store object ownership.
`-tbl` [*`owner`*`.]`*`tableName`*	Displays lock information for the specified table. Requires SELECT privileges or data store object ownership.
`-V | -version`	Prints the release number of **ttXactAdmin** and exits.
`-v` *`verbosity`*	Specifies the verbosity level. One of: 0—does not display the names of the tables for row locks. In this case, **ttXactAdmin** runs faster. 1—(the default) displays the names of the tables for row locks.
`-xact x`*`id`*	Displays information for the specified transaction. The LSNs of the specified transaction are automatically included in the output. Requires SELECT privileges or data store object ownership.
`-xactIdRollback` *`xid`*	Allows you to rollback a transaction. This may be particularly useful for long running transactions. This parameter *`xid`* represents the transaction ID. This stops any currently executing operations on behalf of that transaction and then rolls back the transaction in TimesTen. If there is currently a checkpoint in process when the rollback is requested, TimesTen terminates the checkpoint operation. Requires ADMIN privileges or data store object ownership.

Output **ttXactAdmin** produces the following output:

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 data store made by a single multithreaded process.
TransId	The unique identifier for the transaction used internally by TimesTen. The identifier has two parts. The first part is a relatively small value (less than 2048), used to discriminate between transactions that are active at the same time. The second part is a potentially large value (an unsigned integer), and is used to discriminate between successive uses of the same first part. (The value wraps around if necessary.) Thus, identifiers 4.100 and 4.200 cannot be present at the same time. If 4.100 is seen, and then 4.200, this indicates that transaction 4.100 has completed (committed or rolled back).
TransStatus	Current status of the transaction, one of: `Active` - Active transaction `Committing` - Committing transaction, locks are being released. `Ckpointing` - A transaction doing checkpoint. `Rep-Wait-Return` - Replicated transaction waiting Return Receipt/Commit. `Idle` - A transaction branch currently not accessing data. `Prepared` - Prepared transaction branch. `Heur-Committed` - Heuristically committed transaction branch. `Heur-Aborted` - Heuristically aborted transaction branch. `Propagating` - TimesTen transaction waiting for Oracle to commit.

Resource	The type of the lock being requested: `Row` - Row-level lock. `HashedKey` - A lock held on a key value of a hash index; acquired when an operation requires a hash index to be updated. `Table` - Table-level lock. `EndScan` - End of table or T-tree scan lock. `Database` - Data store-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 except for Table and CompCmd which are shown as decimal values.

Mode	A value used to determine the level of concurrency that the lock provides: S – Shared lock in serializable isolation. Sn – Shared lock in non-serializable isolation. U – Update lock in serializable isolation. Un – Update lock in non-serializable isolation. En – End-of-scan lock for non-serializable isolation. IRC – Intention shared lock in non-serializable isolation. IS – Intention shared lock in serializable isolation IU – Intention update lock in serializable isolation IUn – Intention update lock in non-serializable isolation IX – Intention exclusive lock in serializable isolation IXn – Intention exclusive lock non-serializable isolation SIX – Shared lock with intent to set an exclusive lock in serializable isolation. SIXn – Shared lock with intent to set an exclusive lock non-serializable isolation. X – Exclusive lock. Xn – Exclusive lock in non-serializable isolation. W – Update, insert or delete table lock. XNi – Next lock for inserting into tables or non-unique index NS – Table lock in read-committed isolation that conflicts with all table locks in serializable isolation Lock “0” means the blocker is still in the waiting list.
HMode	The mode in which the competing transaction is holding the lock which the waiting transaction is requesting. See "Mode" in this table for concurrency level descriptions.
RMode	Shows the mode in which the waiting transaction has requested to hold the lock. See "Mode" in this table for concurrency level descriptions.
HolderTransId	The identifier of the transaction with which the waiting transaction is in contention.
Name	The name of the table that the lock is being held on or within.

Example The following command displays all locks in the data store:

```
ttXactAdmin –connstr DSN=demodata
Outstanding locks
```

PID	Con-text	TransId	Trans-Status	Resource	Resour-ceId	Mode	Name
Program File Name: localtest1							
10546	0x118e28	2047.000003	Active	Table	411104	IS	SYS.TABLES
				Table	416480	IXn	TEST1.TAB1
				Row	0x00065ae0	Sn	SYS.TABLES
				Hashed Key	0x69cf9c36	Sn	SYS.TABLES
				Data-base	0x01312d00	IXn IX	
				Row	0x000eebfc	Xn	TEST1.TAB1
Program File Name: /users/smith/demo/XAtest1 XA-XID: 0xbea1-001b238716dc35a7425-64280531947e1657380c5b8d							
1817	0x118e28	2046.000004	Active	Table	416480	IS	TEST1.TAB1
				CompCmd	21662408	S	
				Database	20000000	IS	
				Row	0x000eebf0	Sn	TEST1.TAB1
Program File Name: /users/smith/demo/XAtest2 XA-XID: 0xbea1-001c99476cf9b21e85e1-70657473746f7265506f6f6c							

2731 7	0x118e 28	2045.000 005	Pre- pared	Table	411104	IS	SYS.TABL ES
				Table	416816	IXn	TEST1.TA B2
				Row	0x00065 c30	Sn	SYS.TABL ES
				Data- base	0x01312 d00	IXn	
				Hashed- Key	0x67fe3 852	Sn	SYS.TABL ES
				Row	0x000ef 804	Xn	TEST1.TA B2

Program File Name: /users/smith/demo/Reptest

275 89	0x118 e28	2044.00 0006	Rep-Wait- Return

Awaiting locks

PI D	Con- text	TransId	Resou rce	Resour ceId	RMo de	Holder- TransId	HMo de	Name
Program File Name: /users/smith/demo/XAtest1								
18 17	0x118 e28	2046.00 0004	Row	0x000e ebfc	Sn	2047.000 003	Xn	TEST1. TAB1

The following command displays all locks for transaction 2045.000005:

```
ttXactAdmin –xact 2045.000005 -connstr DSN=demodata

PID     Context   Trans-    1st LSN    Last LSN   Resource   Resour-     Mod   Name
                  Status                                     ceId        e

Program File Name: /users/smith/demo/XAtest2
XA-XID: 0xbea1-001c99476cf9b21e85e1-70657473746f7265506f6f6c

273     0x118     Prepa     0.0116     0.0116     Table      411104      IS    SYS.TA
17      e28       red       404        452                                    BLES

                                                  Table      416816      IXn   TEST1.
                                                                               TAB2

                                                  Row        0x00065     Sn    SYS.TA
                                                             c30                BLES

                                                  Data-      0x01312     IXn
                                                  base       d00

                                                  Hashed     0x67fe3     Sn    SYS.TA
                                                  Key        852                BLES

                                                  Row        0x000ef     Xn    TEST1.
                                                             804                TAB2
```

To display all the connections to the data store:

```
$ ttXactAdmin -connections sample
2006-09-10 10:26:33
/datastore/terry/sample
TimesTen Release 7.0.0.0.0

  ID  PID    Context            Name           Program       State TransID UID
   1 29508   0x00000001001c6680  myconnection  ttIsql         Run   1.23 TERRY
2044 29505   0x0000000100165290 Worker         timestensubd   Run        TERRY
2045 29505   0x00000001001df190 Flusher        timestensubd   Run        TERRY
2046 29505   0x000000010021cc50 Monitor        timestensubd   Run        TERRY
2047 29505   0x0000000100206730 Checkpoint     timestensubd   Run        TERRY
5 connections found
```

Notes: If the transaction specified in the command is not an XA transaction branch but a TimesTen local transaction, no XA-XID are displayed. The XA-XID is a C structure that contains a format identifier, two length fields and a data field. The data field consists of at most two contiguous components: a global transaction identifier (gtrid) and a branch qualifier (bqual). The two length fields specify the number of bytes (1-64) in gtrid and bqual respectively. For more details, refer to

the *X/Open publication: Distributed Transaction Processing: The XA Specification (c193)*.

Under RMode, awaiting transactions are sorted by PID and Context. The listing does not reflect the order of the lock requests.

A lock request with an RMode compatible with the HMode of the lock holder can be waiting because there is another lock request with an incompatible mode ahead of the compatible request in the lock request queue.

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 data store. 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.

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]
[-at dir] [-af file] [-lb] [-headers recs] [-logdir dir]
{-connStr connection_string | DSN | dspath}
```

Options

ttXactLog has the options:

`-af file`	Write autotruncation status to file *file*. Ignored if `-at` not given. Requires ADMIN privileges or data store object ownership.
`-at dir`	Auto truncation mode. Truncates the log file at the first error. Moves the old log files to directory dir and overwrites the file that was in that location. Use only at the request of TimesTen Customer Support, as loss of data may occur. Requires ADMIN privileges or data store object ownership.
`-b blkID`	Restricts log records to those accessing this block, plus any transaction records.
`-connStr` *connection_string*	An ODBC connection string containing the name of the data store, the server name and DSN (if necessary) and any relevant connection attributes.
DSN	The ODBC source name of the data store for which to display the transaction log.

dspath	The fully qualified name of the data store. This is not the DSN associated with the connection but the fully qualified data store path name associated with the data store as specified in the `DataStore=` parameter of the data store's ODBC definition. For example, for a data store consisting of files `/home/payroll/1997.ds0, /home/payroll/1997.ds1` and several log files `/home/payroll/1997.log`*n*, *dspath* is `/home/payroll/1997`.

`-h`
`-help`
`-?`
	Prints a usage message and exits.

`-headers` *records*	Prints one header for every *records* records. A value of 0 disables headers entirely.
`-lb`	Connects to the data store and prints out the log buffer. Contents of the log files are not printed. Requires SELECT privileges or data store object ownership.
lfn.lfo	Log file number (*lfn*) and 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.
`-logdir` *dir*	Specifies the directory where the data store's log files reside. If `-logdir` is not specified, ttXactLog uses the directory path portion of the value supplied in *dspath*.
`-m` *maxChars*	Maximum number of characters printed for BINARY items (for `-v 3`) only (defaults to 1000).
`-r` *recType*	Considers only records of the specified type. This option may be used more than once to specify a list of desired log record types. *recType* is case-sensitive.
`-s`	Prints summary information. Requires SELECT privileges or data store object ownership.
`-t`	Only reads log file tail (from start of last checkpoint log file or, if no checkpoint, the most recent log file).

`-tr` *`dir`*	All records from the one specified by `-l1` onwards are deleted. The original log files are moved to the directory *`dir`*.
`-V | -version`	Prints the release number of **ttXactLog** and exits.
`-v` *`verbosity`*	Specifies the verbosity level. One of: 0—print only summary log information (if `-s` specified). 1—(the default) print log record headers too. 2—print log record bodies too, except long data. 3—print full log records (see `-m` option).

Example
```
ttXactLog -v 3 -m 100 /users/pat/TimesTen/Daily/F112697SS
```

Note This utility is supported only where the TimesTen Data Manager is installed.

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 example:

Example 3.1 The following call tells the optimizer that it should not generate temporary hash indexes when preparing commands:

ODBC

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

JDBC

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

Built-in procedure list

Procedure	See
ttAgingLRUConfig	page 204
ttAgingScheduleNow	page 206
ttApplicationContext	page 207
ttBackupStatus	page 208
ttBlockInfo	page 210
ttBookmark	page 211
ttCacheAWTThresholdGet	page 212
ttCacheAWTThresholdSet	page 213
ttCacheMonitor	page 214
ttCachePolicyGet	page 216
ttCachePolicySet	page 217
ttCachePropagateFlagSet	page 219
ttCacheSqlGet	page 220
ttCacheStart	page 222
ttCacheStop	page 223
ttCacheUidGet	page 224
ttCacheUidPwdSet	page 225
ttCkpt	page 226
ttCkptBlocking	page 228
ttCkptConfig	page 230
ttCkptHistory	page 233
ttCommitLSN	page 237
ttCompact	page 238
ttCompactTS	page 239
ttConfiguration	page 240

Procedure	See
ttXlaBookmarkCreate	page 331
ttXlaBookmarkDelete	page 332
ttXlaSubscribe	page 333
ttXlaUnsubscribe	page 334

ttAgingLRUConfig

Description

Defines the Least Recently Used (LRU) aging attributes for tables and cache groups. To turn on LRU Aging, use **CREATE TABLE** or **ALTER TABLE**.

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

Data is removed if the data store space in-use exceeds the specified threshold values. LRU aging occurs at the table level. With cache group tables, it can only be defined with the root table. LRU aging is not allowed for cache groups with AUTOREFRESH. For those cache groups, use time-based aging.

If no arguments are supplied, this procedure only returns the current settings.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

Syntax

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

Parameters

ttAgingLRUConfig has these optional parameters:

Parameter	Type	Description
lowUsageThreshold	BINARY_FLOAT	Low end of percentage of data store PermSize. The bottom of the threshold range in which LRU aging should be deactivated. Default is 80%.
highUsageThreshold	BINARY_FLOAT	High end of percentage of data store PermSize. The top of the threshold range in which LRU aging should be activated. Default is 90%.
agingCycle	TT_INTEGER	Number of minutes between aging cycles. Default is 1 minute.

Result set

ttAgingLRUConfig returns these results:

Parameter	Type	Description
lowUsageThreshold	BINARY_FLOAT NOT NULL	The current setting for the low end of percentage of data store PermSize.

highUsageThreshold	BINARY_FLOAT NOT NULL	The current setting for the high end of percentage of data store PermSize.
agingCycle	TT_INTEGER NOT NULL	The current setting for the number of minutes between aging cycles.

Example 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, do not supply any of the optional arguments. Assuming the default settings, if you use:

```
Call ttAgingLRUConfig();
```

The procedure returns:

<.8000000, .9000000, 1>

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

Any of the arguments can be omitted. In the case that an argument is omitted, the previous value is retained.

See also "ttAgingScheduleNow" on page 206.
TimesTen Cache Connect to Oracle Guide.

ttAgingScheduleNow

Description

Start the aging process immediately, or at the time desired, if used in an external scheduler, such as a `cron` job. To use this procedure in an external scheduler, the table aging state should be OFF. Aging occurs only once when this procedure when the application calls this procedure. It does not change any aging properties. For tables with aging ON the aging cycle is reset to the time when this procedure is called.

Applications can set aging OFF to disable automatic the aging cycle and use this procedure to kick off aging using the external scheduler.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DELETE privileges.

Syntax

```
ttAgingScheduleNow ('tablename')
```

Parameters

ttAgingScheduleNow has the parameter:

Parameter	Type	Description
tablename	TT_CHAR (61)	The tableName on which aging is to be triggered. If *tableName* is omitted, all tables that have a time-based aging policy are examined and data that is no longer needed is aged out.

Result set

ttAgingScheduleNow returns no results.

Example

```
CALL ttAgingScheduleNow ();
```

See also

"ttAgingLRUConfig" on page 204.
TimesTen Cache Connect to Oracle Guide.

ttApplicationContext

Description Sets application-defined context for the next update record (either and UPDATE or commit) in order to pass application specific data to XLA readers.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires WRITE privileges.

Syntax

```
ttApplicationContext (cmd)
```

Parameters **ttApplicationContext** has the parameter:

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

Result set **ttApplicationContext** returns no results.

Example

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

See also The "XLA Reference" chapter in the *Oracle TimesTen In-Memory Database C Developer's and Reference Guide*.

ttBackupStatus

Description

Returns a single row with information about the current or last backup of the data store. 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.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store since the last first-connect. **1** - A backup is currently in progress. **2** - The last backup completed successfully. **3** - The last backup failed. In this case the error column contains the error code for the failure.
destination	TT_INTEGER	The type of backup taken. The value is NULL when no backup has been taken on the data store. 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 data store. 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 data store.
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 log file number of the backup point. The value is NULL when no backup has been taken on the data store.
backupLFO	TT_INTEGER	The log file offset of the backup point. The value is NULL when no backup has been taken on the data store.
error	TT_INTEGER	In the case of a failed backup, 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 data store.
processId	TT_INTEGER	The ID of the process or daemon performing the backup (if known).

Example

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

Notes

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

Information returned is not persistent across data store startup or shutdown.

ttBlockInfo

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

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

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 data store
FreeBlocks	TT_BIGINT NOT NULL	Total number of free blocks in the data store
FreeBytes	TT_BIGINT NOT NULL	Total size of the free blocks
LargestFree	TT_BIGINT NOT NULL	Size of the largest free block

Example

```
CALL ttBlockInfo;
```

ttBookmark

Description

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

- A log file number, and
- An offset in that log file.

Log file numbers correspond to the file system names given to log files. For example, the log file `SalesData.log29` has the 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.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttBookmark
```

Parameters

ttBookmark has no parameters.

Result set

ttBookmark returns the result set:

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

Example

```
CALL ttBookmark;
```

ttCacheAWTThresholdGet

Description Returns the current log file threshold for data stores that include AWT cache groups.

Syntax

```
ttCacheAWTThresholdGet ()
```

Parameters ttCacheAWTThresholdGet has no parameters.

Result set ttCacheAWTThresholdGet returns the result:

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

Example

```
CALL ttCacheAWTThresholdGet();
```

Note This procedure is available only on platforms that support TimesTen's Cache Connect feature.

See Also "ttCacheAWTThresholdSet" on page 213

ttCacheAWTThresholdSet

Description

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

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

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

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

Syntax

```
ttCacheAWTThresholdSet (threshold)
```

Parameters

ttCacheAWTThresholdSet has the parameter:

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

Result set

ttCacheAWTThresholdSet returns no results.

Example

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

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

Note

This procedure is available only on platforms that support TimesTen's Cache Connect feature.

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

See Also

"ttCacheAWTThresholdGet" on page 212

ttCacheMonitor

Description Monitors the performance and settings of the specified cache group.

Syntax `ttCacheMonitor (featureName, cacheGroupName, tblName)`

Parameters **ttCacheMonitor** has the optional parameters:

Parameter	Type	Description
featureName	TT_VARCHAR (100) NOT NULL	The name of the cache attribute to return information about. Legal values are: • cgLastRefreshRow • cgLastRefreshDuration • cgTimer • cgAgingInterval • cgMemThreshold
cacheGroupName	TT_VARCHAR (100)	The name of the cache group. Only needed for *cgLastRefreshRow* and *cgLastRefreshDuration*.
tblName	TT_VARCHAR (100)	Must be specified NULL. Reserved for future use.

Result set **ttCacheMonitor** has the return values, based on the indicated parameters:

Column	Type	Value
retValue	TT_INTEGER NOT NULL	Returns information on one of: • **cgLastRefreshRow** - Number of rows updated in the last autorefresh. • **cgLastRefreshDuration** - Duration, in milliseconds, of last autorefresh or asynchronous writethrough. • **cgTimer** - Current aging timer value in seconds • **cgAgingInterval** - Aging interval in number of minutes • **cgMemThreshold** - Percentage of Memory Threshold

Example

```
CALL ttCacheMonitor('cgLastRefreshRow', 'myGroup', NULL);
```

Notes

This procedure is available only on platforms that support TimesTen's Cache Connect feature.

ttCachePolicyGet

Description Returns the current policy used to determine when the TimesTen cache agent for the connected data store should run. The policy can be either **always** or **manual**.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store should run. Valid values are: **always** - specifies that the agent for the data store is always running. This option immediately starts the TimesTen cache agent. When the TimesTen daemon restarts, TimesTen automatically restarts the cache agent **manual** - (the default) specifies that you must manually start the cache agent using either the **ttCacheStart** built-in procedure or the **ttAdmin** `-cacheStart` command. You must explicitly stop the cache agent using either the **ttCacheStop** built-in procedure or the **ttAdmin** `-cacheStop` command.

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

```
CALL ttCachePolicyGet ();
```

Notes This procedure is available only on platforms that support TimesTen's Cache Connect feature.

See Also

"ttCachePolicySet" on page 217
"ttCacheStart" on page 222
"ttCacheStop" on page 223
"ttCacheUidGet" on page 224
"ttCacheUidPwdSet" on page 225
"ttAdmin" on page 75

ttCachePolicySet

Description Defines the policy used to determine when the TimesTen cache agent for the connected data store should run. The policy can be either **always** or **manual**.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store should run. Valid values are: **always** - specifies that the agent for the data store is always running. This option immediately starts the TimesTen cache agent. When the TimesTen daemon restarts, TimesTen automatically restarts the cache agent **manual** - (the default) specifies that you must manually start the cache agent using either the **ttCacheStart** built-in procedure or the **`ttAdmin`** `-cacheStart` command. You must explicitly stop the cache agent using either the **ttCacheStop** built-in procedure or the **`ttAdmin`** `-cacheStop` command.

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 on platforms that support TimesTen's Cache Connect feature.

If you attempt to start the TimesTen cache agent (by changing the policy from **manual** to **always**) for a data store with a relative path, TimesTen looks for the data store relative to where TimesTen Data Manager is running, and fails. For example, on Windows, if you specify the path for the data store 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 data store in the *`install_dir`*`\srv` directory. On UNIX, TimesTen Data Manager looks in `/var/TimesTen/`*`instance/`* directory.

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

See Also

"ttCachePolicyGet" on page 216
"ttCacheStart" on page 222
"ttCacheStop" on page 223
"ttCacheUidGet" on page 224
"ttCacheUidPwdSet" on page 225
"ttAdmin" on page 75

ttCachePropagateFlagSet

Description

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

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires WRITE privileges.

Syntax

```
ttCachePropagateFlagSet (cacheCommitsOn)
```

Parameters

ttCachePropagateFlagSet has the parameter:

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

Result Set

ttCachePropagateFlagSet returns no results.

Notes

This procedure is available only on platforms that support TimesTen's Cache Connect feature.

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

The propagate flag is reset after a commit or rollback.

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

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

ttCacheSqlGet

Description Generates the Oracle SQL statements to install or uninstall Oracle objects for incremental and ASYNCHRONOUS WRITETHROUGH cache groups.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

Syntax `ttCacheSqlGet(`*`feature_name, cache_group_name, install_flag`*`)`

Parameters **ttCacheSqlGet** has these parameters:

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

Result set **ttCacheSqlGet** returns the result set:

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

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

Example

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

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

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

Notes

This procedure is available only on platforms that support TimesTen's Cache Connect feature.

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

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

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

ttCacheStart

Description Starts the TimesTen cache agent for the connected data store.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttCacheStart ()
```

Parameters **ttCacheStart** has no parameters.

Result Set **ttCacheStart** returns no results.

Examples To start the TimesTen cache agent, use:

```
CALL ttCacheStart ();
```

Note This procedure is available only on platforms that support TimesTen's Cache Connect feature.

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

If you attempt to start the TimesTen cache agent (by changing the policy from **manual** to **always**) for a data store with a relative path, TimesTen looks for the data store relative to where the TimesTen Data Manager is running, and fails. For example, on Windows, if you specify the path for the data store 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 data store in the *`install_dir`*`\srv` directory. On UNIX, the TimesTen Data Manager looks in `/var/TimesTen/`*`instance/`* directory.

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

See Also

"ttCachePolicySet" on page 217
"ttCacheStop" on page 223
"ttCacheUidPwdSet" on page 225
"ttCacheUidGet" on page 224
"ttAdmin" on page 75

ttCacheStop

Description Stops the TimesTen Oracle for the connected data store.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttCacheStop (timeout)
```

Parameters **ttCacheStop** has the parameter:

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

Result Set **ttCacheStop** returns no results.

Examples To stop the TimesTen cache agent, use.

```
CALL ttCacheStop();
```

Notes This procedure is available only on platforms that support TimesTen's Cache Connect feature.

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

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

See Also

"ttCachePolicySet" on page 217
"ttCacheStart" on page 222
"ttCacheUidPwdSet" on page 225
"ttCacheUidGet" on page 224
"ttAdmin" on page 75

ttCacheUidGet

Description Gets the cache administration user ID and password for the data store. If the cache administration user ID and password has not been set for the data store, then **ttCacheUidGet** returns NULL.

Syntax

```
ttCacheUidGet ( )
```

Parameters `ttCacheUidGet` has no parameters.

Result set **ttCacheUidGet** returns the results:

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

Example

```
CALL ttCacheUidGet();
```

Note This procedure is available only on platforms that support TimesTen's Cache Connect feature.

See Also

"ttAdmin" on page 75
"ttCacheUidPwdSet" on page 225

ttCacheUidPwdSet

Description

Sets the cache administration user ID and password. The cache administration user ID and password only need to be specified once for each new data store. The cache administration password can be changed at any time.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

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

Parameters

ttCacheUidPwdSet has these parameters:

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

Result set

ttCacheUidPwdSet returns no results.

Example

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

Notes

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

This procedure is available only on platforms that support TimesTen's Cache Connect feature.

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

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

Also

"ttAdmin" on page 75
"ttCacheUidGet" on page 224

ttCkpt

Description

Performs a non-blocking checkpoint when logging to disk is enabled. If logging is off, this procedure does a blocking checkpoint. The blocking checkpoints are described in **ttCkptBlocking**. See "ttCkptBlocking" on page 228 A checkpoint operation is used to make a record of the current state of the data store on disk and to purge log files. A non-blocking checkpoint does not require any locks on the data store.

Applications should checkpoint data stores periodically either by setting the background checkpointing attributes (**CkptFrequency** and **CkptLogVolume**) or by explicitly calling this procedure.

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 will wait for the running backup or checkpoint to finish.
- If a backup or checkpoint is running and you try to do a checkpoint, it will not wait—it will return an error right away.

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

This procedure can be called asynchronously to any other application running on the data store.

When a data store crashes, and the checkpoints on disk are non-blocking checkpoints,TimesTen uses the log to recover. For this reason, non-blocking checkpoints cannot be used when logging to disk is turned off.

Because this procedure is non-blocking, the timeout and retries parameters are used only for applications that are not logging to disk, when this procedure does a blocking checkpoint. Please see the description of **ttCkptBlocking** for their use.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store lock before timing out. The value of *timeout* can be between 0 and one million, inclusively. If not specified, it defaults to infinity (the checkpoint never times out). The procedure only uses this value when the checkpoint is blocking, that is, when logging to disk is not enabled.
retries	TT_INTEGER	The number of times that **ttCkpt** should attempt to get a data store lock, if timeouts occur. The value of *retries* can be between 0 and 10, inclusive. If not specified, it defaults to 0. The procedure only uses this value when the checkpoint is blocking, that is, when logging to disk is not enabled.

Result set **ttCkpt** returns no results.

Example

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

Notes For a description of checkpoints, see the "Transaction Management and Recovery" chapter of the *TimesTen Developer's Guide*.

See also

"ttCkptBlocking" on page 228
"ttCkptConfig" on page 230
"ttCkptHistory" on page 233

ttCkptBlocking

Description

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

When this procedure is called, TimesTen performs a blocking checkpoint when the current transaction is committed or rolled back. If, at that time, other transactions are in progress, the checkpointing connection waits until the other transactions have committed or rolled back. While the checkpoint connection is waiting, any other new transactions that want to start form a queue behind the checkpointing transaction. As a result, if any transaction is long-running, it may cause many other transactions to be held up. So, this blocking checkpoint should be used with caution. To perform a non-blocking checkpoint, use the **ttCkpt** procedure. (See "ttCkpt" on page 226)

No log is needed to recover when blocking checkpoints are used. Hence this type of checkpoint can be used even when logging to disk is disabled. TimesTen uses the log, if present, to bring the data store up to date after recovery.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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

Result set

ttCkptBlocking returns no results.

Example

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

Notes

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

For more information on checkpoints, see the "Transaction Management and Recovery" chapter of the *TimesTen Developer's Guide*.

See also

"ttCkpt" on page 226
"ttCkptConfig" on page 230
"ttCkptHistory" on page 233

ttCkptConfig

Description

Changes the configuration of 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 data store 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.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

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

Parameters

ttCkptConfig has these parameters:

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

Result set

ttCkptConfig returns these results:

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

Examples

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

```
CALL ttCkptConfig;
< 600, 32 >
```

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

```
CALL ttCkptConfig(0);
< 0, 32 >
```

To stop the background checkpointer from initiating checkpoints, use:

```
CALL ttCkptConfig(NULL, 0);
< 0, 0 >
```

To set the background checkpointer configuration to initiate a checkpoint every 600 seconds or to checkpoint when the log reaches 32 megabytes (whichever comes first), use:

```
CALL ttCkptConfig(600, 32);
< 600, 32 >
```

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 will wait for the running backup or checkpoint to finish.
- If a backup or checkpoint is running and you try to do a checkpoint, it will not wait—it will return an error right away.

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

See also

"CkptFrequency" on page 23
"CkptLogVolume" on page 24
"ttCkpt" on page 226.
"ttCkptHistory" on page 233
"ttCkptHistory" on page 233

ttCkptHistory

Description Returns information about the last eight checkpoints of any type taken by any agent.

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 data store creation and at last disconnect. **Blocking -** Transaction-consistent checkpoint. **Fuzzy -** Non-blocking checkpoint. The background checkpointer performs this type if possible. **None -** For Temporary data stores, which have no checkpoint files.
status	TT_CHAR (16) NOT NULL	Result status of the checkpoint operation. Value is one of: **In Progress -** The checkpoint is currently in progress. Only the most recent result row can have this status. **Completed -** The checkpoint completed successfully. **Failed -** The checkpoint failed. Only the most recent result row can have this status. In this case the *error* column indicates the reason for the failure.

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 data store. For a shared data store, the final disconnect checkpoint is taken by the subdaemon.
error	TT_INTEGER	In the case of a **Failed** checkpoint, this column indicates the reason for the failure. The value is one of the TimesTen error numbers.
ckptFileNum	TT_INTEGER NOT NULL	The data store 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 log file number of the checkpoint log record.
ckptLFO	TT_INTEGER	The log file offset of the checkpoint log record.
blksTotal	TT_BIGINT	The number of permanent blocks currently allocated in the data store. These blocks are subject to consideration for checkpointing.
bytesTotal	TT_BIGINT	The number of bytes occupied by *blksTotal*.
blksInUse	TT_BIGINT	Of *blksTotal*, the number of blocks currently in use.
bytesInUse	TT_BIGINT	The number of bytes occupied by *blksInUse*.
blksDirty	TT_BIGINT	The number of dirty blocks written by this checkpoint.
bytesDirty	TT_BIGINT	The number of bytes occupied by *blksDirty.*

bytesWritten	TT_BIGINT	The total number of bytes written by this checkpoint.
Percent_Complete	TT_INTEGER	If there is an in-progress checkpoint, indicates the percentage of the checkpoint that has been completed. If no checkpoint is in-progress, the value is NULL. The returned value is calculated by comparing the block ID of the last-written block against the data store's PermSize. The value does not necessarily indicate the precise time remaining to complete the checkpoint, although it does give some indication of the remaining time needed to complete the disk write. The field shows only the progress of the writing of dirty blocks and does not include additional bookkeeping at the end of the checkpoint.

Examples

```
CALL ttCkptHistory;

< 2005-03-15 16:15:36.000000, 1753-01-01 00:00:00.000000, Fuzzy, In
Progress, Checkpointer, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0 >

< 2005-03-15 16:14:36.000000, 2005-03-15 16:14:37.000000, Fuzzy,
Completed, Checkpointer, 0, 1, 0, 357768, 22, 2097152, 21, 980720,
11, 748960, 822496 >

< 2005-03-15 16:13:36.000000, 2005-03-15 16:13:38.000000, Blocking,
Completed, User, 0, 0, 0, 357712, 22, 2097152, 21, 980720, 10, 683800,
1506296 >

< 2005-03-15 16:13:00.000000, 2005-03-15 16:13:01.000000, Static,
Completed, Subdaemon, 0, 1, 0, 357616, 22, 2097152, 21, 980720, 10,
683800, 683800 >

< 2005-03-15 16:12:56.000000, 2005-03-15 16:12:57.000000, Fuzzy,
Completed, User, 0, 0, 0, 357520, 22, 2097152, 21, 980720, 2, 92736,
134368 >

< 2005-03-15 16:12:36.000000, 2005-03-15 16:12:38.000000, Fuzzy,
Completed, Checkpointer, 0, 1, 0, 357424, 22, 2097152, 21, 980720,
10, 683800, 789728 >

< 2005-03-15 16:11:36.000000, 2005-03-15 16:11:39.000000, Static,
Completed, User, 0, 0, 0, 357368, 22, 2097152, 21, 980720, 22, 2097152,
980864 >

< 2005-03-15 16:11:31.000000, 2005-03-15 16:11:36.000000, Static,
Completed, User, 0, 1, 0, 357312, 22, 2097152, 21, 980720, 22, 2097152,
980864 >


CALL ttCkptHistory;

< 2005-03-15 17:03:43.000000, 2005-03-15 17:03:43.000000, Fuzzy,
Failed, Checkpointer, 847, 1, -1, -1, 0, 0, 0, 0, 0, 0, 0 >

< 2005-03-15 17:02:43.000000, 2005-03-15 17:02:44.000000, Static,
Completed, Subdaemon, 0, 0, 0, 362704, 22, 2097152, 21, 980720, 10,
683800, 683800 >
```

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" on page 226
"ttCkptBlocking" on page 228

ttCommitLSN

Description Returns a single result row containing the log sequence number (LSN) of the commit record of the connection's last non-read-only transaction.

Syntax

```
ttCommitLSN
```

Parameters **ttCommitLSN** has no parameters.

Result set **ttCommitLSN** returns the result set:

Column	Type	Description
CommitLFN	TT_INTEGER	Log file number of commit log record
CommitLFO	TT_INTEGER	Log file offset of the commit log record

Example

```
CALL ttCommitLSN;
```

Note A successful invocation of **ttCommitLSN** always returns exactly one result row. If a connection has never executed a transaction that updates the data store, or if the data store is in no logging mode, then **ttCommitLSN** returns a row containing -1 for both `CommitLFN` and `CommitLFO`.

ttCompact

Description

Compacts the data store. **ttCompact** compacts both the permanent and temporary data partitions.

Call **ttCompact** periodically to reorganize the internal structure of a data store. It may also be useful to call **ttCompact** when the application receives out of memory errors because the cause of the problem may be data store fragmentation.

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**. To eliminate this type of fragmentation, consider using **ttMigrate** -rebuild. (See "ttMigrate" on page 136).

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttCompact ( )
```

Parameters

ttCompact has no parameters.

Result set

ttCompact returns no results.

Example

```
CALL ttCompact;
```

Note

Compacting data does not modify result addresses.

See also

"ttCompactTS" on page 239

ttCompactTS

Description

ttCompactTS is similar to **ttCompact**, except that **ttCompactTS** may be used to compact a small fraction of the data store, while **ttCompact** compacts the entire data store. **ttCompactTS** is a time-sliced version of **ttCompact**. **ttCompactTS** iterates through all the blocks in the data store compacting the quantum specified each time. ttCompactTS called repeatedly ultimately has the same effect as a call to **ttCompact**. When a sweep is completed, the value of the DS_COMPACTS field in the MONITOR table is incremented.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttCompactTS (quantum)
```

Parameters

ttCompactTS has the parameter:

Parameter	Type	Description
quantum	TT_INTEGER NOT NULL	A non-zero positive INTEGER that specifies the number of data blocks a **ttCompactTS** should compact. Each quantum corresponds to one data block.

Result set

ttCompactTS returns no results.

Example

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

Note

Compacting data does not modify result addresses.

See also

"ttCompact" on page 238

ttConfiguration

Description

Queries the ODBC connection string to determine the values of most connection attribute for the current connection.

The connection attributes that are returned include:

CkptFrequency	CkptLogVolume
CkptRate	ConnectionCharacterSet
ConnectionName	Connections
DataBaseCharacterSet	DataStore
DurableCommits	GroupRestrict
LockLevel	LockWait
LogBuffSize	LogDir
LogFileSize	LogFlushMethod
LogPurge	Logging
MemoryLock	NLS_LENGTH_SEMANTICS
NLS_NCHAR_CONV_EXCP	NLS_SORT
OracleID	PassThrough
PermSize	PermWarnThreshold
Preallocate	PrivateCommands
RACCallback	SQLQueryTimeout
Temporary	TempSize
TempWarnThreshold	TypeMode
UID	

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires CONNECT privileges.

Syntax

```
ttConfiguration ()
```

Parameters

ttConfiguration has no parameters.

Result set

ttConfiguration returns the result set:

Column	Type	Description
ParameterName	TT_VARCHAR (30) NOT NULL	The names of the connection attributes specified in the connection string, returned in alphabetical order.
ParameterValue	TT_VARCHAR (1024)	The values of the connection attributes specified in the connection string.

Example

```
CALL ttConfiguration ();
```

Notes

Only the attributes listed above are returned by this procedure. Client driver attributes are not returned by this procedure.

See also

Chapter 1, “Data Store Attributes” in this guide.

ttContext

Description Returns the context value of the current connection as a BINARY(8) value. The context can be used to correlate a unique connection to a data store from the list of connections presented by the **ttStatus** utility and the **ttDataStoreStatus** built-in procedure.

Syntax

```
ttContext
```

Parameters **ttContext** has no parameters.

Result set **ttContext** returns the result set:

Column	Type	Description
CONTEXT	BINARY(8)	Current connection's context value.

Example

```
CALL ttContext;
```

Note The context value numbers are unique only within a process. The context value number is not unique within the entire data store. Therefore you may see the same context value number for different processes.

See also

"ttDataStoreStatus" on page 243
"ttStatus" on page 176

ttDataStoreStatus

Description

Returns the list of processes connected to a data store. If the DATASTORE parameter is specified as NULL, then the status of all active data stores is returned.

The result set is similar to the printed output of the **ttStatus** utility.

Syntax

```
ttDataStoreStatus (dataStore)
```

Parameters

ttDataStoreStatus has the parameter:

Parameter	Type	Description
dataStore	TT_VARCHAR (128)	Full path name of desired data store or NULL for all data stores.

Result set

ttDataStoreStatus returns the result set:

Column	Type	Description
dataStore	TT_VARCHAR (128) NOT NULL	Full path name of data store.
PID	TT_INTEGER NOT NULL	Process ID.
Context	BINARY(8) NOT NULL	Context value of connection.
conType	TT_CHAR (16) NOT NULL	Type of process connected. The result can be one of the following: **application** - an ordinary application is connected. **replication** - a replication agent is connected. **subdaemon** - a subdaemon is connected. **oracleagent** - an cache agent is connected.
ShmID	TT_VARCHAR (260) NOT NULL	A printable version of the shared memory ID that the data store occupies.

connectionName	TT_CHAR (30) NOT NULL	The symbolic name of the data store connection.
connID	TT_INTEGER NOT NULL	The numeric ID of the data store connection

Example

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

See also

"ttContext" on page 242
"ttStatus" on page 176

ttDurableCommit

Description

Indicates that the current transaction should be made durable when it is committed. It only has an effect if the application is connected to the data store with **DurableCommits** disabled (see "DurableCommits" on page 42) and logging to disk enabled (see "Logging" on page 33).

Calling **ttDurableCommit** also makes the current transaction and any previously committed non-durable transactions durable. There is no effect on other transactions that are committed subsequent to calling **ttDurableCommit**. **ttDurableCommit** does not commit transactions. The application must do the commit, for example with a call to **SQLTransact**.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttDurableCommit ( )
```

Parameters

ttDurableCommit has no parameters.

Result set

ttDurableCommit returns no results.

Example

```
CALL ttDurableCommit;
```

Note

Some controllers or drivers may only write data into cache memory in the controller or may write to disk some time after the operating system is told that the write is done. In these cases, a power failure may mean that some information you thought was durably committed does not survive the power failure. To avoid this loss of data, configure your disk to write all the way to the recording media before reporting completion or you can use an Uninterruptable Power Supply.

ttHostNameGet

Description

Returns the current local host for the current data store. The value returned is only for the current session, it is not a system-wide setting and does not persist after the current session has been disconnected.

This procedure can be used to check whether a particular store name in a scheme refers to the current host. This can be helpful when configuring replication schemes.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires CONNECT privileges.

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 data store. If a default has not been supplied then the current hostname is returned.

Example

```
CALL ttHostNameGet ();
```

See also

."ttHostNameSet" on page 247

Chapter 4, "Setting Up a Replicated System" in the *TimesTen to TimesTen Replication Guide*

ttHostNameSet

Description

Sets the default local host for the current data store. The value is only used in the current session, it is not a system-wide 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 machine on which the script is currently being run. This procedure allows an application to set a default host name for the current session that can be used by Replication DDL processing whenever there is a need to establish the name of the current host.
Access Control

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires CONNECT privileges.

Syntax

```
ttHostnameSet (hostName)
```

Parameters

ttHostNameSet has the parameter:

Parameter	Type	Description
hostName	TT_VARCHAR (200)	The required default name for the local machine. If NULL is supplied the default value is cleared.

Result set

ttHostNameSet returns no results.

Example

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

Note

The legal value can an IP address string. It cannot be set to "`localhost`", "127.0.0.1" or "::1". You cannot set the default hostname to a value that is different to the local hostname used in an existing replication scheme.

See also

"ttHostNameGet" on page 246

Chapter 4, "Setting Up a Replicated System" in the *TimesTen to TimesTen Replication Guide*

ttLockLevel

Description

Changes the lock level between row-level and data store-level locking on the *next* transaction and for all subsequent transactions for this connection. Applications can change the lock level again by calling **ttLockLevel** once more. The initial value depends on the **LockLevel** attribute (see "LockLevel" on page 44).

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store-level locking.

Result set

ttLockLevel returns no results.

Example

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

Note

This procedure does not affect the current transaction.

Row-level locking is required when caching Oracle tables.

This procedure must be called from within a transaction. It has the effect of setting the locking level for subsequent transactions for the connection that invoked it. The new lock level does not affect the current transaction. It takes effect at the beginning of the next transaction.

See also

"ttLockWait" on page 249

ttLockWait

Description

Allows an application to change the lock timeout interval of the current connection. The change takes effect immediately and applies to all subsequent statements in the current transaction, as well as 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.

If **AutoCommit** is off at the time that **ttLockWait** is called, the application must commit the transaction in which **ttLockWait** is executed.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttLockWait (seconds)
```

Parameters

ttLockWait has the required parameters:

Parameter	Type	Description
seconds	NUMBER (8,1) NOT NULL	Number of seconds to wait for a lock when there is contention on it. You can also specify fractions of a second. Values between 0.0 and 1000000.0 inclusive are accepted.

Result set

ttLockWait returns no results.

Examples

To indicate a six second lock wait, use:

```
CALL ttLockWait (6);
```

To indicate a tenth of a second lock wait, use:

```
CALL ttLockWait (0.1);
```

Notes

When a lock is not immediately available to a TimesTen transaction, it waits a predetermined amount of time to try and get the lock. After that it times out the lock request and returns TimesTen error 6003 to the application. By default, TimesTen uses a value of 10 seconds for lock timeouts.

Of special interest is the lock time-out interval value of 0. If that value is specified, transactions do not wait for any unavailable locks. If the lock is not available, the request returns with TimesTen error 6003.

See also "ttLockLevel" on page 248
"LockWait" on page 45

ttLogBufPrint

Description A debugging procedure that prints the contents of the in-memory log buffer to standard output.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

Syntax

```
ttLogBufPrint (verbose, maxLen)
```

Parameters **ttLogBufPrint** has these parameters:

Parameter	Type	Description
verbose	TT_INTEGER	The value of *verbose* indicates how much information should be printed: 0: do not print anything 1: just headers 2: main body 3: everything Default value is 2.
maxLen	TT_INTEGER	The value of *maxLen* gives the maximum length to use for buffering the extra formatted part of each log record that gets printed. Default value is 0.

Result set **ttLogBufPrint** returns no results.

Example

```
CALL ttLogBufPrint(2,0);
```

ttLogHolds

Description Retrieves information about log holds, including those created on behalf of incremental backups, replication peers, persistent XLA subscribers, XA, long-running transactions and checkpoints. This procedure can help diagnose situations where it appears that checkpoint operations are not purging all unneeded log files.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

Syntax

```
ttLogHolds ()
```

Parameters **ttLogHolds** has no parameters.

Result set **ttLogHolds** returns the result set:

Column	Type	Description
HoldLFN	TT_INTEGER NOT NULL	Returns the log file number of the hold.
HoldLFO	TT_INTEGER NOT NULL	Returns the log file offset of the hold
type	TT_CHAR (30) NOT NULL	Returns the type of hold, one of: **Checkpoint** **Replication** **Backup** **XLA** **Long-Running Transaction** **Long-Running XA Transaction**
description	TT_VARCHAR (1024) NOT NULL	Describes the type-specific object for which the hold was created. Each description corresponds with the Type returned. Descriptions are one of: - The name of the checkpoint file. - The name of the replication subscriber. - The backup path. - The name of the persistent XLA subscription and the process ID of the last process to open it, if it is open. - The XA XID (transaction ID) of the XA transaction. - The TimesTen transaction ID of the long-running transaction.

Example

```
CALL ttLogHolds();
```

Output

```
< 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 >
```

ttOptClearStats

Description

Clears (deletes) 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 SQL Reference Guide*.)

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

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, statistics are cleared for all of the current user's tables in the data store.
invalidate	TT_INTEGER	0 (no) or 1 (yes). If *invalidate* is 1, all commands that reference the affected tables are re-prepared automatically when they are re-executed, including commands prepared by other users. If *invalidate* is 0, the statistics are not considered modified and existing commands are not reprepared.

Result set

ttOptClearStats returns no results.

Example

```
CALL ttOptClearStats ( 'SALLY.ACCTS', 1 );
```

Clears the statistics for the SALLY.ACCTS table and reprepares all commands that affect the ACCTS table.

```
CALL ttOptClearStats();
```

Clears the statistics for all of the current user's tables and reprepares all commands that affect these tables.

```
CALL ttOptClearStats('', 0);
```

Clears the statistics for all of the current user's tables without repreparing commands that reference these tables.

See also “ttOptEstimateStats” on page 256.
“ttOptSetColIntvlStats” on page 262.
“ttOptSetFlag” on page 266.
“ttOptSetOrder” on page 273.
“ttOptSetTblStats” on page 276.
“ttOptUpdateStats” on page 280.
“ttOptUseIndex” on page 283.

ttOptEstimateStats

Description

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'`).

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

Syntax

```
ttOptEstimateStats (tblName, invalidate, sampleStr)
```

Parameters

ttOptEstimateStats has these parameters:

Parameter	Type	Description
tblName	TT_CHAR (61)	Name of an application table. Can include table owner. If *tblName* is the empty string, statistics are estimated for all of the current user's tables in the data store.
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.

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 sub-optimal execution plans. For "good" distributions of data, a 10 percent selection is a good choice for computing statistics quickly without sacrificing plan accuracy. If the number of rows specified is sufficiently large or the table in question is sufficiently small, to improve performance TimesTen computes exact statistics anyway on all columns that have a length of 2,048 bytes or less. For example, the only difference between

```
ttOptEstimateStats ('ACCTS', 1, '100 PERCENT' )
```

and

ttOptUpdateStats `( 'ACCTS', 1 )`

is that the former does not compute statistics for long columns.

The statistics are stored in the TBL_STATS and COL_STATS system tables.

For performance reasons, TimesTen does not hold a lock on tables or rows when computing statistics. However, computing statistics can still slow performance. Estimating statistics generally provides better performance than computing exact statistics.

See also

"ttOptSetColIntvlStats" on page 262.
"ttOptSetFlag" on page 266.
"ttOptSetOrder" on page 273.
"ttOptSetTblStats" on page 276.
"ttOptUpdateStats" on page 280.
"ttOptUseIndex" on page 283.

ttOptGetColStats

Description

Returns statistics information in text format.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

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.
colName	TT_CHAR (61)	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 (61)	Name of the table.
colName	TT_CHAR (61)	Name of the column.
stats	TT_VARCHAR (409600) NOT NULL	Statistics in text form.

Examples

```
CALL ttOptGetColStats ();
```

< T1 , X1, (2, 10, 10, 100 (,4, 40, 10 ,1, 10, 5) ,(4, 20, 20 ,11, 20, 15))>

See also

"ttOptSetColStats" on page 264
"ttOptSetColIntvlStats" on page 262

ttOptGetFlag

Description

Returns the optimizer flag settings for the current transaction.
The results are returned as a result set that can be retrieved using the ODBC **SQLFetch** function or the JDBC ResultSet.getXXX method, just like the result of a SQL SELECT statement. Applications can request the value of a specific optimizer flag by passing the flag name to **ttOptGetFlag**. Alternatively, applications can request the values of all the optimizer flags by passing NULL. The optimizer flags and their meanings are described under the **ttOptSetFlag** built-in procedure. (See "ttOptSetFlag" on page 266.)

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

Syntax

```
ttOptGetFlag (flagName)
```

Parameters

ttOptGetFlag has the parameter:

Parameter	Type	Description
flagName	TT_CHAR (32)	Name of the flag whose value is to be returned. If NULL is passed, then the values of all flags are returned.

Result set

ttOptGetFlag returns the result set:

Column	Type	Description
flagName	TT_VARCHAR (32) NOT NULL	Name of the flag. See "ttOptSetFlag" on page 266 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" on page 266.

ttOptGetMaxCmdFreeListCnt

Description

Returns the size of the SQL compiled command cache. To reset the size of the cache, use ttOptSetMaxCmdFreeListCnt for regular tables and ttOptSetMaxPriCmdFreeListCnt for materialized views.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

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.

Example

```
CALL ttOptGetMaxCmdFreeListCnt ( );
```

Notes

See also

"ttOptSetMaxCmdFreeListCnt" on page 271.
"ttOptSetMaxPriCmdFreeListCnt" on page 272.

ttOptGetOrder

Description Returns a single-row result set containing the join order for the current transaction. This result set can be retrieved using the ODBC **SQLFetch** function or the JDBC ResultSet.getXXX method, just like the result of a SQL SELECT statement. Join orders are described under the **ttOptSetOrder** built-in procedure.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

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" on page 273.

ttOptSetColIntvlStats

Description

Modifies the statistics for the specified columns with interval information. This procedure allows an application to set statistics manually rather than have TimesTen automatically compute them. This feature is useful for preparing commands before the data has been inserted or for seeing how table characteristics can affect the choice of execution plan. This procedure modifies the relevant row(s) in the COL_STATS system table.

Because this procedure can be used before any data are in the table, the values specified do not need to bear any relation to the actual values, although some basic validity checking is performed.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

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.
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.

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, modVal),`
		`  /* information for interval 2 */`
		`...)`

Result set

ttOptSetColIntvlStats returns no results.

Example To set the following statistics for column t1.x1:

- Two intervals
- Integer type
- 10 rows with null value
- 10 unique value
- 100 rows
- Interval 1 (4 unique values besides the most frequently occurring value, 40 rows with values other than most frequently occurring value, 10 rows with most frequently occurring value, min = 1, max = 10, mod = 5)
- Interval 2 (4 unique values besides the most frequently occurring value, 20 rows with values other than most frequently occurring, 20 rows with most frequently occurring value, min = 11, max = 20, mod = 15)

Use the statement:

```
CALLttOptSetColIntvlStats('t1', 'x1', 1, (2, 10, 10, 100,
(4, 40, 10, 1, 10, 5), (4, 20, 20, 11, 20, 15)));
```

Notes The minimum and maximum values in the interval need to be given 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” on page 256
“ttOptGetColStats” on page 258
“ttOptSetColStats” on page 264
“ttOptSetTblStats” on page 276.
“ttOptUpdateStats” on page 280.

ttOptSetColStats

Description

Modifies the statistics for the specified columns. This procedure allows an application to set statistics manually rather than have TimesTen automatically compute them. This feature is useful for preparing commands before the data has been inserted or for seeing how table characteristics can affect the choice of execution plan. This procedure modifies the relevant row(s) in the COL_STATS system table.

Because this procedure can be used before any data are in the table, the values specified do not need to bear any relation to the actual values, although some basic validity checking is performed.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

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.
colName	TT_CHAR (30) NOT NULL	Name of a column in that table.
numUniq	TT_INTEGER NOT NULL	Number of unique values in the column.
minVal	VARBINARY(1024) NOT NULL	Minimum value in the column (possibly truncated).
maxVal	VARBINARY(1024) NOT NULL	Maximum value in the column (possibly truncated).

invalidate	TT_INTEGER	0 (no) or 1 (yes). If *invalidate* is 1, all commands that reference the affected tables are automatically prepared again when re-executed. This includes commands prepared by other users. If *invalidate* is 0, the statistics are not considered to have been modified and existing commands are not reprepared.
numNull	TT_INTEGER	Indicates the total number of NULLs in the column.

Result set **ttOptSetColStats** returns no results.

Example

```
CALL ttOptSetColStats ( 'SALLY.ACCTS', 'BALANCE',
                        400, 0x00001388, 0x000186A0, 1, 0);
```

Notes The minimum and maximum values need to be given 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" on page 256
"ttOptGetColStats" on page 258
"ttOptSetColIntvlStats" on page 262
"ttOptSetTblStats" on page 276.
"ttOptUpdateStats" on page 280.

ttOptSetFlag

Description

Allows applications to alter the generation of execution plans by the TimesTen query optimizer. It sets flags to enable or disable the use of various access methods. The changes made by this call take effect during preparation of statements and affect all subsequent calls to the ODBC functions **SQLPrepare and SQLExecDirect** or the JDBC methods **Connection.prepareCall** and **Statement.execute** in the current transaction. All optimizer flags are reset to their default values when the transaction has been committed or rolled back. If optimizer flags are set while **AutoCommit** is on, they are ignored because each statement is executed within its own transaction.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

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	0 (disable) or 1 (enable).

Optimizer Flags

When setting the optimizer flags, use the following character strings, which are not case sensitive:

`FirstRow`	Enable or disable 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.
`GenPlan`	Enable or disable the creation of entries in the PLAN table for the rest of the transaction. (See "Generating the plan" in the *TimesTen Operations Guide*.)
`Hash`	Allow or disallow the use of existing hash indexes in indexed table scans.

`MergeJoin`	Refers to a method for joining two tables. Available when the input streams for the join are sorted. An example is when a T-tree index scan is used.
`NestedLoop`	Refers to a common way of joining two tables.
`PassThrough`	Temporarily changes the pass-through level for Cache Connect applications. The passthrough level can be set at any time and takes effect immediately. Legal values for this flag are: **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, ROLLBACK, SAVEPOINT, and TimesTen built-in procedures that set or get optimizer flags, are passed through. At the end of the transaction, the pass-through level is reset to the original value.
`RowLock`	Allow or disallow the optimizer to consider using row locks.
`Scan`	Refers to full table scans.
`Rowid`	Allow or disallow the use of Row IDs.
`TmpHash`	Allow or disallow the use of a temporary hash scan. This is an index that is created during execution for use in evaluating the statement. Though index creation is time-consuming, it can save time when evaluating join predicates.
`TblLock`	Allow or disallow the optimizer to consider using table locks.
`TmpTable`	Stores intermediate results into a temporary table. This operation is sometimes chosen to avoid repeated evaluation of predicates in join queries or sometimes just to allow faster scans of intermediate results in joins.

`TmpTtree`	Performs a temporary T-tree scan. Can also be used so that values are sorted for a merge join. Though index creation is time-consuming, it can save time when evaluating join predicates.
`Ttree`	Allow or disallow the use of existing T-Tree indexes in indexed table scans.
`TransparentLoad`	Allows or disallows loading of Oracle data to a cache group during a SELECT operation performed on the cache group for the current transaction. TransparentLoad is ignored if the PassThrough value is set to 3, because all operations are performed on the Oracle database. Legal values for this flag are: **0** - Oracle data is not loaded to the cache group when a SELECT operation occurs. (default) **1** - Oracle data is loaded to the cache group when a SELECT operation occurs. In this mode, no error or warning is returned if there is **2** - Oracle data is loaded to the cache group when a SELECT operation occurs. This mode returns an error at compilation or execute time if the SELECT cannot use transparent load. The SELECT will be executed based on the data available in TimesTen only. If Access Control is enabled for your TimesTen instance, INSERT privilege is required on the tables for which you set this flag.

In addition, the string `AllFlags` can be used to refer to all optimizer flags, and the string `Default` can be used to refer to the default flags. `Default` excludes the `GenPlan` flag but includes all other optimizer flags.

Flag Description

The value of each flag can be 1 or 0:

- If 1, the operation is enabled.
- If 0, the operation is disabled unless absolutely necessary.
- Initially, all the flag values *except* `GenPlan` are 1 (all operations are permitted).

For example, an application can prevent the optimizer from choosing a plan that stores intermediate results:

```
ttOptSetFlag ( 'TmpTable', 0 )
```

Similarly, an application can specify a preference for `MergeJoin`:

```
ttOptSetFlag ( 'NestedLoop', 0 )
```

In the second example, the optimizer may still choose a nested loop join if a merge join is impossible (for example, if there is no merge-join predicate). Similarly, the optimizer may occasionally not be able to satisfy an application request to avoid table scans (when the Scan flag is set to 0).

You cannot specify that a particular operation is prohibited only at a certain step of a plan or that a particular join method always be done between two specific tables. Similarly, there is no way to specify that certain indexes be used or that a hash index be used to evaluate a specific predicate. Each operation is either fully permitted or fully restricted.

When a command is prepared, the current optimizer flags, index hints and join order are maintained in the structure of the compiled form of the command and are used if the command is ever reprepared by the system. See Chapter 9, "The TimesTen Query Optimizer" in *Oracle TimesTen In-Memory Database Operations Guide* for an example of reprepared statements.

If both RowLock and TblLock are disabled, TimesTen uses row-locking. If both RowLock and TblLock are enabled, TimesTen uses the locking scheme that is most likely to have better performance:

TblLock status	RowLock status	Effect on the optimizer
Disabled	Disabled	Use row-level locking.
Enabled	Disabled	Use table-level locking.
Disabled	Enabled	Use row-level locking.
Enabled	Enabled	Optimizer chooses row-level or table-level locking.

In general, table-level locking is useful when a query accesses a significant portion of the rows of a table and/or when there are very few concurrent transactions accessing the table.

Result set **ttOptSetFlag** returns no results.

Example

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

See also

"ttOptEstimateStats" on page 256.
"ttOptGetFlag" on page 259.
"ttOptGetOrder" on page 261.
"ttOptSetColIntvlStats" on page 262.
"ttOptSetOrder" on page 273.

“ttOptSetTblStats” on page 276.
“ttOptUpdateStats” on page 280.
“ttOptUseIndex” on page 283.

ttOptSetMaxCmdFreeListCnt

Description

Sets the size of the regular SQL compiled command cache. To get the current setting use the ttOptGetMaxCmdFreeListCnt procedure.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttOptSetMaxCmdFreeListCnt (maxCnt)
```

Parameters

ttOptSetMaxCmdFreeListCnt has the required parameter:

Parameter	Type	Description
maxCnt	TT_INTEGER NOT NULL	The size of the SQL compiled command cache.

Result set

ttOptSetMaxCmdFreeListCnt returns no results.

Example

```
CALL ttOptSetMaxCmdFreeListCnt (40);
```

See also

"ttLogBufPrint" on page 251.
"ttOptEstimateStats" on page 256.

ttOptSetMaxPriCmdFreeListCnt

Description

Sets the size of the compiled command cache for commands that do materialized view maintenance.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttOptSetMaxCmdFreeListCnt ()
```

Parameters

ttOptSetMaxPriCmdFreeListCnt has the required parameter:

Parameter	Type	Description
maxCnt	TT_INTEGER NOT NULL	The size of the SQL compiled command cache.

Result set

ttOptSetMaxPriCmdFreeListCnt returns no results.

Example

```
CALL ttOptSetMaxPriCmdFreeListCnt ( );
```

See also

"ttLogBufPrint" on page 251.
"ttOptEstimateStats" on page 256

ttOptSetOrder

Description Specifies the order in which tables should be joined by the optimizer. The character string is a list of correlation names referenced in the query or a subquery, separated by spaces (*not* commas). The table listed first is scanned first by the plan. (It is outermost in a nested loop join, for example.) A correlation name is a shortcut or alias for a qualified table name.

Syntax

```
ttOptSetOrder (joinOrder)
```

Parameters **ttOptSetOrder** has the required parameter:

Parameter	Type	Description
joinOrder	TT_VARCHAR (1024)	List of space-separated table correlation names. If an owner is required to distinguish the table name, use a table correlation name. If the *joinOrder* is not specified the query optimizer reverts to its default behavior.

Result set **ttOptSetOrder** returns no results.

Example 3.2

```
CALL ttOptSetOrder ('EMPS DEPTS ACCTS');
```

Use the correlation name instead of the actual table name when specifying the join order.

Example 3.3 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 un-shipped order:

```
SELECT CUSTOMERS.NAME
FROM   CUSTOMERS
WHERE  EXISTS (SELECT 1
               FROM   ORDERS
               WHERE  CUSTOMERS.ID = ORDERS.CUSTID
               AND    ORDER.STATUS ='UN-SHIPPED');
```

Notes The string length is limited to 1,024 bytes. If a string exceeds this length, it is truncated and a warning is issued.

When correlation names referenced in subqueries are in included in the order, TimesTen may internally change the isolation mode.

When a command is prepared, the current optimizer flags, index hints, and join order are maintained in the structure of the compiled form of the command and are used if the command is ever reprepared by the system. See the "TimesTen Query Optimizer" Chapter in the *TimesTen Developer's Guide* for an example of reprepared statements.

The changes made by this call take effect immediately and affect all subsequent calls to the ODBC function **SQLPrepare** or the JDBC method **Connection.prepareCall** in the current transaction. The query optimizer reverts to its default behavior for subsequent transactions.

The tables referenced by a query must exactly match the names given if the join order is to be used (the comparisons are not case sensitive). A complete ordering must be specified; there is no mechanism for specifying partial orders. If the query has a subquery then the join order should also reference the correlation names in the subquery. In essence, the join order should reference all the correlation names referenced in the query. The TimesTen optimizer internally implements a subquery as a special kind of join query with a GROUP BY. For the join order to be applicable it should reference all the correlation names. If there is a discrepancy, a warning is issued and the specified join order is ignored completely. Here are some examples:

Example 3.4 If an application makes the call:

```
ttOptSetOrder('DEPTS EMPS ACCTS');
```

the optimizer is prevented from executing a join between DEPTS and ACCTS when evaluating the number of employees working on a specific account:

```
SELECT COUNT(DISTINCT EMPS.ID)
FROM   ACCTS, DEPTS, EMPS
WHERE  ACCTS.DEPTS = DEPTS.ID
AND    EMPS.DEPTS = DEPTS.ID
AND    ACCTS.NUM = :AcctNum
```

If the application does not reset the join order and tries to prepare a command that does not reference each of the three tables (and no others), the optimizer issues warning number 965. The specified join order is not applicable. TimesTen considers valid join orders and ignores the specified join order when preparing the command.

See also
"ttLogBufPrint" on page 251.
"ttOptEstimateStats" on page 256.
"ttOptGetFlag" on page 259.
"ttOptGetOrder" on page 261.
"ttOptSetColIntvlStats" on page 262.
"ttOptSetFlag" on page 266.

“ttOptSetTblStats” on page 276.
“ttOptUpdateStats” on page 280.
“ttOptUseIndex” on page 283.

ttOptSetTblStats

Description

Modifies the statistics for the specified table. This procedure allows an application to set statistics explicitly rather than have TimesTen automatically compute them.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

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.
numRows	TT_INTEGER NOT NULL	Number of rows in the table.
invalidate	TT_INTEGER	0 (no) or 1 (yes). If *invalidate* is 1, all commands that reference the affected tables are automatically prepared again when re-executed, including commands prepared by other users. If *invalidate* is 0, the statistics are not considered to have been modified and existing commands are not reprepared.

Result set

ttOptSetTblStats returns no results.

Example

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

Note

This feature is useful for preparing commands before the data has been inserted or for seeing how table size can affect the choice of an execution plan. Because the command can be used before any data are in the table, the values specified do not need to bear any relation to the actual values.
This procedure modifies the relevant row(s) in the TBL_STATS system table (see "SYS.TBL_STATS" in *Oracle TimesTen In-Memory Database SQL Reference Guide*).

See also

“ttOptEstimateStats” on page 256.
“ttOptGetFlag” on page 259.
“ttOptGetOrder” on page 261.

“ttOptSetColIntvlStats” on page 262.
“ttOptSetFlag” on page 266.
“ttOptSetOrder” on page 273.
“ttOptUpdateStats” on page 280.
“ttOptUseIndex” on page 283.

ttOptShowJoinOrder

Description

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, the **ttOptSetFlag** `('ShowJoinOrder', 1)` or set the **ttIsql** "ShowJoinOrder" command to ON (1) must be set first in the same transaction. AUTOCOMMIT must be off when using either of these commands. The join order is represented by table names.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

Syntax

```
ttOptShowJoinOrder ( )
```

Parameters

ttOptShowJoinOrder has no parameters.

Result set

ttOptShowJoinOrder returns the result:

Column	Type	Description
tblName	TT VARCHAR (4096) NOT NULL	Table names, including owner name quantifiers and correlation name for each table if specified. Table names are returned in parentheses.

Example

```
>AUTOCOMMIT 0;
> CALL ttOptSetFlag ('ShowJoinOrder', 1);
>PREPARE SELECT * FROM t1;
>CALL ttOptShowJoinOrder();
>( T1 )
```

Note

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" on page 256.
"ttOptGetFlag" on page 259.
"ttOptGetOrder" on page 261.
"ttOptSetColIntvlStats" on page 262.
"ttOptSetFlag" on page 266.
"ttOptSetOrder" on page 273.
"ttOptSetTblStats" on page 276.

“ttOptUpdateStats” on page 280.
“ttOptUseIndex” on page 283.

ttOptUpdateStats

Description

Updates the statistics for the specified table. TimesTen looks at the data stored 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.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

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 of the current user's tables are updated.

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.
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 t-tree index exists on the column. If a t-tree index does not exist, only single interval statistics are collected. **1** - Do not collect complete interval statistics. Only single interval statistics are collected. See "Notes" for more information.

Result set **ttOptUpdateStats** returns no results.

Example

```
CALL ttOptUpdateStats ( 'ACCTS', 1 );
```

Updates the ACCTS table and causes all commands that reference the ACCTS table to be re-prepared when they are next executed.

```
CALL ttOptUpdateStats('', 1);
```

Updates all of 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 of 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 NULLs in the column
- Total number of NON NULL UNIQUE values in the column
- Total number of rows in the table
- Interval information, each interval contains:
  - The minimum value
  - The maximum value
  - The most frequently occurring value
  - The number of times the most frequent value occurred
  - The number of rows that have different values than the most frequent value
  - The number of unique values besides the most frequent value

Collection of complete interval statistics requires the data to be sorted.

If complete interval statistics are not selected, then statistics are collected by treating the entire distribution as a single interval.

For performance reasons, TimesTen does not hold a lock on tables or rows when computing statistics. However, computing statistics can still slow performance. Estimating statistics generally provides better performance than computing exact statistics. See "ttOptEstimateStats" on page 256 for information on estimating statistics.

See also

"ttOptEstimateStats" on page 256.
"ttOptGetColStats" on page 258.
"ttOptSetColStats" on page 264.
"ttOptSetColIntvlStats" on page 262.
"ttOptSetTblStats" on page 276.
"ttOptUpdateStats" on page 280.

ttOptUseIndex

Description

Allows applications to alter the generation of execution plans by the TimesTen query optimizer. It allows applications to disable the use of a set of indexes or enable the consideration of only a set of indexes for each correlation used in a query. Enabling the consideration of an index does not guarantee that the plan generated uses the index. Depending on the estimated cost, the optimizer might choose to use a serialization scan or a materialization scan to access the associated correlation if these scans resulted in a better plan than the ones that use the specified index.

The changes made by this call take effect immediately and affect all subsequent calls to the ODBC functions **SQLPrepare** and **SQLExecDirect** or the JDBC methods **Connection.prepareCall** and **Statement.execute** in the current transaction until the applications explicitly issue a call to clear it. The setting is cleared whenever a new transaction is started.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

Syntax

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

Parameters

ttOptUseIndex has a single string parameter of type VARCHAR(1024) with these components:

Component	Description
IndexName	The name of the user-defined index or '_TMPTTREE' for temporary T-tree index or '_TMPHASH' for temporary hash index. If index name is omitted, the setting applies to all indexes of the specified correlation.
CorrelationName	The correlation name of the table. If a table is defined with a correlation name in the FROM clause, use this correlation name instead of the table name when specifying the index hint for this table. If correlation name is omitted for an entry, the setting affects all tables with the specified index name.
0 | 1	Disables(0) or enables (1) the use of the index specified by IndexName.

Result set

ttOptUseIndex returns no results.

Examples

```
CALL ttOptUseIndex('"3456"."1234", t1, 0');
CALL ttOptUseIndex('data1.i1, data1.t1, 0');
CALL ttOptUseIndex('i1, t1, 0');
```

Note

If **ttOptUseIndex** is called without a parameter or with a NULL value, TimesTen clears the previous index hint.

See also

“ttOptEstimateStats” on page 256.
“ttOptGetFlag” on page 259.
“ttOptGetOrder” on page 261.
“ttOptSetColIntvlStats” on page 262.
“ttOptSetFlag” on page 266.
“ttOptSetOrder” on page 273.
“ttOptSetTblStats” on page 276.
“ttOptUpdateStats” on page 280.
“ttOptUseIndex” on page 283.

ttRamPolicyGet

Description

Returns the replication restart policy used to determine when the TimesTen Replication Agent for the connected data store should run. The policy can be either *always*, *manual*, or *norestart*.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

`ttRamPolicyGet(`*`ramPolicy`*`)`

Parameters

ttRamPolicyGet has this parameter.

Result Set

ttRamPolicyGet returns the results:

Parameter	Type	Description
ramPolicy	TT_VARCHAR (10) NOT NULL	The policy used to determine when the data store is loaded into system RAM. Valid values are: **always** - specifies that the data store should remain in system RAM all the time. **manual** - specifies that the data store is only to be loaded in system RAM when explicitly loaded by the user, using the **ttAdmin** `-ramLoad` command. **inUse** - specifies that the data store is only loaded in system RAM when in use (i.e.:when applications are connected). This option cannot be used with temporary data stores. TimesTen only allows a temporary data store to be loaded into RAM manually. Trying to set the policy generates a warning.
ramGrace	TT_INTEGER	If the ramPolicy is inUse, this field reports the number of seconds the data store is kept in RAM after the last application has disconnected. Otherwise, this field is NULL.

Examples To view the RAM policy, use:

```
CALL ttRamPolicyGet();
```

See Also “ttAdmin” on page 75.

ttRamPolicySet

Description

Defines the replication restart policy used to determine when the TimesTen Replication Agent for the connected data store should run. The policy can be either *always*, *manual*, or *norestart*.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttRepPolicy(ramPolicy, ramGrace)
```

Parameters

ttRepPolicy has this parameter:

Parameter	Type	Description
ramPolicy	TT_VARCHAR (10) NOT NULL	The policy used to determine when the data store is loaded into system RAM. Valid values are: **always** - specifies that the data store should remain in system RAM all the time. **manual** - specifies that the data store is only to be loaded in system RAM when explicitly loaded by the user, using the **ttAdmin** `-repStart` command. **inUse** - specifies that the data store is only loaded in system RAM when in use (i.e.:when applications are connected). This option cannot be used with temporary data stores. TimesTen only allows a temporary data store to be loaded into RAM manually. Trying to set the policy generates a warning.
ramGrace	TT_INTEGER	Sets the number of seconds the data store 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 data store into RAM to be inUse and for the data store to kept in RAM for 10 seconds after the last application has disconnected, use:

```
CALL ttRamPolicySet('inUse, 10');
```

See Also "ttAdmin" on page 75.

ttRepDeactivate

Description

Changes the state of the active data store in an active standby pair from ACTIVE to IDLE. Use the **ttRepDeactivate** procedure when reversing the roles of the master data stores in an active standby pair.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privilege.

Syntax

```
ttRepDeactivate( )
```

Parameters

ttRepDeactivate has no parameters.

Result Set

ttRepDeactivate returns no results.

Example

To deactivate the active data store in an active standby pair, use:

```
CALL ttRepDeactivate();
```

See Also

“ttRepTransmitSet” on page 317
“ttRepTransmitGet” on page 316
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStateSave” on page 299
“ttRepStateSet” on page 301
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepStateSet” on page 301

ttReplicationStatus

Description Returns the status of one or more replication peer data stores.

Syntax `ttReplicationStatus(`*`receiver, hostname`*`)`

Parameters **ttReplicationStatus** has the optional parameters:

Parameter	Type	Description
receiver	TT_VARCHAR (200)	Subscriber of interest or NULL for all subscribers. If the parameter is provided, then it names a replication subscriber about which information is sought. If the parameter is not provided, then information on replication subscribers defined for the current data store is returned.
hostname	TT_VARCHAR (200)	The host name of one or more stores that are configured to receive updates from the executing store; if NULL, then receiving stores are identified by subscriber alone. If both *receiver* and *hostname* are NULL, then all receiving stores are selected.

Result set **ttReplicationStatus** returns the result set:

Column	Type	Description
subscriber	TT_VARCHAR (200) NOT NULL	Subscriber name.
hostName	TT_VARCHAR (200) NOT NULL	Host name.
port	TT_INTEGER NOT NULL	Defined port number.

pState	TT_CHAR (10) NOT NULL	Peer state. The values of the result column are: **start** - replication is enabled to this peer. **pause** - replication is temporarily paused to this peer. **stop** - replication updates are NOT being collected for this peer. **failed** - replication to a subscriber is considered failed because the threshold limit (log data) has been exceeded. This state is set by the system.
logs	TT_INTEGER NOT NULL	Number of log files held for this peer.
lastMsg	INTEGER	Seconds since last interaction or NULL.
replicationName	TT_CHAR (30) NOT NULL	Name of replication scheme.
replicationOwner	TT_CHAR (30) NOT NULL	Owner of replication scheme.

Example

```
CALL ttReplicationStatus('System8');
```

Notes

If the *receiver* parameter is not NULL, only the status of the given receiver is returned. If the *receiver* parameter is NULL, the status of all subscribers is returned.

This procedure is supported only for TimesTen Data Manager ODBC applications. It is not supported for TimesTen Client or JDBC applications.

See Also

“ttRepDeactivate” on page 289
“ttRepTransmitSet” on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttRepPolicySet” on page 294
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSyncGet” on page 309
“ttRepSyncSet” on page 312

ttRepPolicyGet

Description

Returns the replication restart policy used to determine when the TimesTen Replication Agent for the connected data store should run. The policy can be either *always*, *manual*, or *norestart*.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttRepPolicyGet()
```

Parameters

ttRepPolicyGet has no parameters.

Result Set

ttRepPolicyGet returns the results:

Parameter	Type	Description
repPolicy	TT_VARCHAR (10) NOT NULL	The policy used to determine when the TimesTen Replication Agent for the data store should run. Valid values are: **always** - specifies that the replication agent for the data store 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 data store 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” on page 289
“ttRepTransmitSet” on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStart” on page 296
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepSyncGet” on page 309
“ttRepSyncSet” on page 312

ttRepPolicySet

Description

Defines the replication restart policy used to determine when the TimesTen Replication Agent for the connected data store should run. The policy can be either *always*, *manual*, or *norestart*.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store should run. Valid values are: **always** - specifies that the replication agent for the data store 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 data store is not to be restarted after a failure.

Result Set

ttRepPolicySet returns no results.

Examples

To set the policy for TimesTen Replication Agent to always, use:

```
CALL ttRepPolicySet('always');
```

See Also

“ttRepDeactivate” on page 289
“ttRepTransmitSet” on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicyGet” on page 292
“ttRepStart” on page 296
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepSyncGet” on page 309
“ttRepSyncSet” on page 312

ttRepStart

Description Starts the TimesTen Replication Agent for the connected data store.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttRepStart()
```

Parameters **ttRepStart** has no parameters.

Result Set **ttRepStart** returns no results.

Examples To start the replication agent, use:

```
CALL ttRepStart();
```

Note The replication agent does not start if the data store 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 data store-level locking (**LockLevel**=1).

See Also

“ttRepDeactivate” on page 289
“ttRepTransmitGet” on page 316
“ttRepTransmitSet” on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepSyncGet” on page 309
“ttRepSyncSet” on page 312

ttRepStateGet

Description Indicates the current replication state of a data store in an active standby pair.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privilege.

Syntax

```
ttRepStateGet( )
```

Parameters **ttRepStateGet** has no parameters.

Result Set **ttRepStateGet** returns the result:

Column	Type	Description
state	TT_VARCHAR (20) NOT NULL	The current replication state of the data store. One of: ACTIVE - The data store is currently the active master data store. Applications may update its replicated tables STANDBY - The data store is the standby master data store. Applications may only update its non-replicated tables. FAILED - The data store is a failed master data store. No updates are replicated to it. IDLE - The data store 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 whether the standby data store in an active standby pair has moved from the IDLE to the STANDBY state, use:

```
CALL ttRepStateGet();
< STANDBY >
```

See Also

"ttRepDeactivate" on page 289
"ttRepTransmitSet" on page 317

“ttRepDeactivate” on page 289
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStateSave” on page 299
“ttRepStateSet” on page 301
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepStateSet” on page 301

ttRepStateSave

Description

Saves the state of a remote peer data store in an active standby pair to the currently connected datastore. Currently, may only be used to indicate to the active data store that the standby data store, *storeName* on *hostName*, has failed, and that all updates on the active data store should be replicated directly to the read-only subscribers.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store. May only be specified as FAILED in this release. Recording that a standby data store has failed indicates that all replicated updates are to be sent directly from the active data store to the read-only subscribers.
storeName	TT_VARCHAR (200) NOT NULL	Name of the data store for which the state is indicated.
hostName	TT_VARCHAR (200) NOT NULL	Name of the host where the data store resides.

Result Set

ttRepStateSave returns no results.

Examples

To indicate to the active data store that the standby data store `standby` on host `backup1` has failed, use:

```
ttRepStateSave('FAILED', 'standby', 'backup1');
```

See Also

“ttRepDeactivate” on page 289
“ttRepTransmitSet” on page 317
“ttRepDeactivate” on page 289
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStateGet” on page 297

ttRepStateSet

Description

Sets the replication state of a data store in an active standby pair replication scheme. Currently, **ttRepStateSet** may only be used to set the state of a data store 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 data store has had a CREATE ACTIVE STANDBY PAIR command executed and no failures have occurred since.
- A data store is currently in the STANDBY state, and the other datastore in the active standby pair has had its state changed from ACTIVE to IDLE using the **ttRepDeactivate** procedure.
- A data store has just recovered from the local transaction log and was in the ACTIVE state before it went down.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privilege.

Syntax

```
ttRepStateSet (state)
```

Parameters

ttRepStateSet has the parameter:

Parameter	Type	Description
state	VARCHAR (20) NOT NULL	The replication state of the data store. Must be ACTIVE, in this release. Setting a store to ACTIVE designates it as the active data store in an active standby pair.

Result Set

ttRepStateSet returns no results.

Examples

To set the replication state of the data store to ACTIVE, use:

```
CALL ttRepStateSet('ACTIVE');
```

See Also

“ttRepDeactivate” on page 289
“ttRepTransmitSet” on page 317
“ttRepDeactivate” on page 289
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStateGet” on page 297
“ttRepStateSave” on page 299
“ttRepStop” on page 303

ttRepStop

Description Stops the TimesTen Replication Agent for the connected data store.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store-level locking (**LockLevel**=1).

See Also

“ttRepDeactivate” on page 289
“ttRepTransmitSet” on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guidee*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStart” on page 296
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepSyncGet” on page 309
“ttRepSyncSet” on page 312

ttRepSubscriberStateSet

Description

Changes a replicating subscriber's state with respect to the executing master store.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store whose state is to be set. May be NULL to indicate all stores on host *subscriberHostName*.
subscriberHostName	TT_VARCHAR (200)	The subscriber's host. May be NULL to indicate all hosts of subscribing peers.
newStateCode	TT_INTEGER	An integer code representing the specified subscriber's new state: 0/NULL - started 1 - paused 2 - stopped All other state codes are disallowed. (This procedure cannot set a subscriber state to "failed.")

Result Set **ttRepSubscriberStateSet** returns no results.

Examples For the replication scheme named REPL.REPSCHEME, the following directs the master data store to set the state of the subscriber data store (SUBSCRIBERDS ON SYSTEM1) to Stop (2):

```
CALL ttRepSubscriberStateSet('REPSCHEME', 'REPL', 'SUBSCRIBERDS',
                             'SYSTEM1', 2);
```

To direct the master data store 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" on page 289
"ttRepTransmitSet" on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer's and Reference Guide*
"ttReplicationStatus" on page 290
"ttRepPolicySet" on page 294
"ttRepStart" on page 296
"ttRepStop" on page 303
"ttRepSubscriberWait" on page 306
"ttRepSyncGet" on page 309
"ttRepSyncSet" on page 312

ttRepSubscriberWait

Description

Causes the master or transmitting store to wait for a subscriber to catch up to the write LSN at the time this procedure is called. The **ttRepSubscriberWait** procedure ensures that all updates committed up until the time of the procedure call have been transmitted to the subscriber, *subscriberStoreName*, and that the subscriber has acknowledged the updates have been durably committed at the subscriber data store.

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** function should not be used when an urgent response is required. Instead, you should use the return receipt service.

Note: If this procedure is called after all write transaction activity is quiesced at a store (there are no active transactions and no transactions have started), it may take a 60 seconds or longer before the subscriber sends the acknowledgement that all updates have been durably committed at the subscriber.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 data store 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 with respect to the write LSN at the time this procedure was called. TimesTen returns `0x01` if not enough time has been granted.

Example

If there is one defined replication scheme, to direct the transmitting data store to wait ten minutes for subscriber REP on SERVER2 to catch up, use:

```
CALL ttRepSubscriberWait( , , 'REP' , 'SERVER2', 600 );
```

Note

This procedure is supported only for TimesTen Data Manager ODBC applications. It is not supported for TimesTen Client or JDBC applications.

See Also

"ttRepDeactivate" on page 289
"ttRepTransmitSet" on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer's and Reference Guide*
"ttReplicationStatus" on page 290
"ttRepPolicySet" on page 294
"ttRepStart" on page 296

“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSyncGet” on page 309
“ttRepSyncSet” on page 312

ttRepSyncGet

Description 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.

Syntax

```
ttRepSyncGet ()
```

Parameters **ttRepSyncGet** has no parameters.

Result set **ttRepSyncGet** returns the result set:

Column	Type	Description
requestReturn	BINARY(1)	0 - Don't wait for return notification configured with the RETURN RECEIPT BY REQUEST or RETURN TWOSAFE BY REQUEST option; this value is the default. 1 - Wait for the return notification. Commit resets this attribute to its default value of 0 ("off").

returnWait	TT_INTEGER	Specifies the number of seconds to wait for return service acknowledgement. The default value is 10 seconds. A value of '0' means that there is no wait time. This attribute persists across transaction boundaries and applies to all RETURN services independent of the BY REQUEST option.
localAction	TT_INTEGER	The current LOCAL ACTION configuration for RETURN services. **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, with the exception that the application is not able to update any replicated tables. The application may only reissue the COMMIT. This is the default. **COMMIT** -- When the COMMIT times out, the transaction is committed locally. No more operations are possible on this transaction, and the replicated data stores diverge. This attribute persists across transactions and for the life of the connection.

Example To retrieve the caller's *requestReturn* value, use:

```
SQLCHAR  requestReturn[1];
SQLINTEGER  len;
rc = SQLExecDirect( hstmt
                , (SQLCHAR *) "{CALL ttRepSyncGet( NULL )}"
                , SQL_NTS )
rc = SQLBindCol ( hstmt
                , /* ColumnNumber */ 1
                , /* TargetType */   SQL_C_BINARY )
                , /* TargetValuePtr */requestReturn
                ,./* BufferLength */sizeof requestReturn
                , /* StrLen_ */&len );
rc = SQLFetch( hstmt );
if ( requestReturn[0] ) {
…
}
```

Note When called within a stand-alone transaction, **ttRepSyncGet** always returns the default value for *requestReturn*.

ttRepSyncGet may be called at any point within a transaction in which it is used to request the BY REQUEST return service for that transaction.

If you call **ttRepSyncGet** in a transaction that does not, in fact, update any RETURN RECEIPT BY REQUEST or RETURN TWOSAFE BY REQUEST replication elements, the call has no external effect.

See Also

“ttRepDeactivate” on page 289
“ttRepTransmitSet” on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStart” on page 296
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepSyncSet” on page 312

ttRepSyncSet

Description

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.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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 acknowledgement. The default value is 10 seconds. A value of '0' means there is no wait time. This timeout value overrides the value set by the RETURN WAIT TIME attribute in the CREATE REPLICATION or ALTER REPLICATION statement. The timeout set by this parameter persists across transaction boundaries and applies to all return services independent of the BY REQUEST option.
localAction	TT_INTEGER	Action to be performed in the event the subscriber is unable to acknowledge commit of the transaction within the timeout period specified by returnWait. This parameter can only be used for return twosafe transactions. Set to NULL when using the RETURN service. **1 --** NO ACTION. On timeout, the process recovery commits the transaction. This is equivalent to a forced commit. **2 --** COMMIT. On timeout, the commit function writes a COMMIT log record and effectively ends the transaction locally. No more operations are possible on the same transaction.

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” on page 289
“ttRepTransmitSet” on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStart” on page 296
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepSyncGet” on page 309

ttRepSyncSubscriberStatus

Description

Queries a subscriber data store 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 data store with the specified name on the specified host. You can specify only the *storename*. However, an error is generated if the replication scheme contains more than one subscriber with the same name on different hosts.

Syntax

```
ttRepSyncSubscriberStatus (storeName, hostName)
```

Parameters

ttRepSyncSubscriberStatus has these parameters:

Parameter	Type	Description
storeName	TT_VARCHAR (200) NOT NULL	The name of the subscribing data store to be queried.
hostName	TT_VARCHAR (200)	The host name of one or more stores that are configured to receive updates from the executing store; if NULL, then receiving stores are identified by receiver alone. If both *receiver* and *hostname* are NULL, then all receiving stores are selected.

Result set

ttRepSyncSubscriberStatus returns:

Parameter	Type	Description
disabled	TT_INTEGER	Value is either: ‘1’, if the return service has been disabled on the subscriber data store. ‘0’ if the return service is still enabled on the subscriber data store.

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

Returns the status of transmission of updates to subscribers for the current transaction.

The corresponding **ttRepTransmitSet** built-in procedure allows you to stop transmission of updates to subscribers for the length of a transaction.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttRepTransmitGet()
```

Parameters

ttRepTransmitGet has no parameters.

Result Set

ttRepTransmitGet returns the result:

Parameter	Type	Description
transmit	TT_INTEGER NOT NULL	**0** - Updates are not being transmitted to any subscribers for the remainder of the transaction on the connection. **1**- Updates are being transmitted to subscribers on the connection. (default)

Result Set

ttRepTransmitGet returns no results.

Example

To return the transmit status on the active data store in an active standby pair, use:

```
CALL ttRepTransmitGet();
```

See Also

“ttRepDeactivate” on page 289
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStateSave” on page 299
“ttRepStateSet” on page 301
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306
“ttRepStateSet” on page 301
“ttRepTransmitSet” on page 317

ttRepTransmitSet

Description

Stops subsequent updates on the connection it is executed in from being replicated to any subscriber.

This procedure should be used with care since it could easily lead to transactional inconsistency of remote stores if partial transactions are replicated. If updates are disallowed from getting replicated, the subscriber stores diverge from the master store.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

`ttRepTransmitSet(`*`transmit`*`)`

Parameters

ttRepTransmitSet has the parameter:

Parameter	Type	Description
transmit	TT_INTEGER NOT NULL	When set to 1 updates are transmitted to subscribers on the connection after the built-in is executed to replicate. (default) When set to 0 updates are not transmitted to any subscribers for the remainder of the transaction in which this call was issued, as well as on the connection that issued it.

Result Set

ttRepTransmitSet returns no results.

Example

To deactivate the active data store in an active standby pair, use:

```
CALL ttRepDisallow(0);
```

See Also

“ttRepDeactivate” on page 289
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer’s and Reference Guide*
“ttReplicationStatus” on page 290
“ttRepPolicySet” on page 294
“ttRepStateSave” on page 299
“ttRepStateSet” on page 301
“ttRepStop” on page 303
“ttRepSubscriberStateSet” on page 304
“ttRepSubscriberWait” on page 306

“ttRepStateSet” on page 301
“ttRepTransmitGet” on page 316

ttRepXactStatus

Description

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 in the case of a RETURN TWOSAFE replication transaction that has begun, but not completed, commit processing.

Note: The *error* parameter is only returned for RETURN TWOSAFE transactions.

Syntax

```
ttRepXactStatus (token)
```

Parameters

ttRepXactStatus has the parameter:

Parameter	Type	Description
token	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 data store 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 data store. 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" on page 289
"ttRepTransmitSet" on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer's and Reference Guide*
"ttReplicationStatus" on page 290
"ttRepPolicySet" on page 294
"ttRepStart" on page 296
"ttRepStop" on page 303
"ttRepSubscriberStateSet" on page 304
"ttRepSubscriberWait" on page 306
"ttRepSyncGet" on page 309
"ttRepSyncSet" on page 312
"ttRepXactTokenGet" on page 321

ttRepXactTokenGet

Description

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.

Syntax

```
ttRepXactTokenGet (type)
```

Parameters

ttRepXactTokenGet has these parameters:

Parameter	Type	Description
type	TT_CHAR (2) NOT NULL	The type of transaction desired: **'RR'** -- Return receipt. **'R2'** -- Return twosafe.

Result set

ttRepXactTokenGet returns the result set:

Column	Type	Description
token	VARBINARY (10000)	A varbinary token used to represent the transaction desired.

See Also

"ttRepDeactivate" on page 289
"ttRepTransmitSet" on page 317
"ttRepDuplicateEx" in the *Oracle TimesTen In-Memory Database C Developer's and Reference Guide*
"ttReplicationStatus" on page 290
"ttRepPolicySet" on page 294
"ttRepStart" on page 296
"ttRepStop" on page 303
"ttRepSubscriberStateSet" on page 304
"ttRepSubscriberWait" on page 306
"ttRepSyncGet" on page 309
"ttRepSyncSet" on page 312
"ttRepXactStatus" on page 319

ttSetUserColumnID

Description

This procedure can be used to explicitly set 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.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

Syntax

```
ttSetUserColumnID (tblName, colName, repID)
```

Parameters

ttSetUserColumnID has these parameters:

Parameter	Type	Description
tblName	TT_CHAR(61) NOT NULL	Table name.
colName	TT_CHAR(30) NOT NULL	Column name.
repID	TT_INTEGER NOT NULL	Integer identifier.

Result set

ttSetUserColumnID returns no results.

Example

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

See also

ttSetUserTableID and the *TimesTen to TimesTen Replication Guide*.

ttSetUserTableID

Description

This procedure can be used to explicitly set 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.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires DDL privileges.

Syntax

```
ttSetUserTableID (tblName, repID)
```

Parameters

ttSetUserTableID has these parameters:

Parameter	Type	Description
tblName	TT_CHAR (61) NOT NULL	Table name.
repID	BINARY(8) NOT NULL	Integer identifier.

Result set

ttSetUserTableID returns no results.

Example

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

See also

ttSetUserColumnID and the *TimesTen to TimesTen Replication Guide*.

ttSize

Description

This procedure estimates the size of a table. 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 scan of the table can be avoided by specifying a non-NULL *frac* value, which should be between 0 and 1. This value is used to estimate the average size of varying-length columns. The maximum size of each varying-length column is multiplied by the *frac* value to compute the estimated average size of VARBINARY or VARCHAR columns. If the *frac* parameter is not given, the existing rows in the table are scanned and the average length of the varying-length columns in the existing rows is used. If *frac* is omitted and the table has no rows in it, then *frac* is assumed to have the value 0.5.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires SELECT privileges.

Syntax

```
ttSize(tblName, nRows, frac)
```

Parameters

ttSize has these parameters:

Parameter	Type	Description
tblName	TT_CHAR(61) NOT NULL	Name of an application table. Can include table owner. This parameter is required.
nRows	TT_INTEGER	Number of rows to estimate in a table. This parameter is optional.
frac	BINARY_DOUBLE	Estimated average fraction of VARBINARY or VARCHAR column sizes. This parameter is optional.

Result set

ttSize returns the result set:

Column	Type	Description
size	BINARY_DOUBLE NOT NULL	Estimated size of the table, in bytes.

Examples

```
CALL ttSize('ACCTS', 1000000, NULL);
CALL ttSize('ACCTS', 30000, 0.8);
CALL ttSize('SALES.FORECAST', NULL, NULL);
```

When using **ttSize**, you must first execute the command and then fetch the results. For example:

ODBC

```
double size;
SQLLEN len;

rc = SQLExecDirect(hstmt, "call ttSize('SalesData', 250000,
0.75)", SQL_NTS);
rc = SQLBindColumn(hstmt, 1, SQL_C_DOUBLE, &size, sizeof double,
&len);
rc = SQLFetch(hstmt);
rc = SQLFreeStmt(hstmt, SQL_CLOSE);
```

JDBC

```
. . . . . .
String URL="jdbc:timesten:MyDataStore";
Connection con;
double tblSize=0;
. . . . . .
con = DriverManager.getConnection(URL);
CallableStatement cStmt = con.prepareCall("
{CALL ttSize('SalesData', 250000, 0.75) }");
if( cStmt.execute() )
    {
      rs=cStmt.getResultSet();
      if (rs.next()) {
        tblSize=rs.getDouble(1);
      }
      rs.close();
    }
cStmt.close();
con.close();

. . . . . .
```

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

ttUserPrivileges

Description

Returns user privileges for TimesTen instances that have Access Control enabled. This procedure returns one result row for each privilege granted to each user.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges, unless the argument in CURRENT_USER or the current user's name.

Syntax

`ttUserPrivileges(`*`username`*`)`

Parameters

ttUserPrivileges has the parameter:

Parameter	Type	Description
username	TT_CHAR (30)	If NULL, all users are returned.

Result set

ttUserPrivileges returns the result set:

Column	Type	Description
username	TT_CHAR (30) NOT NULL	User names of TimesTen instance users.
privilege	TT_VARCHAR (32) NOT NULL	Privilege that has been granted to *username*.

Examples

To generate a list of all TimesTen instance users, use:

```
Command> CALL ttUserPrivileges;
< USER1, ALL >
< PUBLIC, CONNECT>
< PUBLIC, CREATE DATASTORE>
< USER2, SELECT>
```

To generate information on the current user, use the special token CURRENT_USER:

```
Command> CALL ttUserPrivileges(CURRENT_USER);
< USER1, ALL>
```

To generate information on the user "terry," use,

```
Command> CALL ttUserPrivileges('terry');
< terry, SELECT>
```

ttUsers

Description Lists the user names for TimesTen instances that have Access Control enabled.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges, unless the current user's name is the value of the argument.

Syntax

```
ttUsers(username)
```

Parameters **ttUsers** has the parameter:

Parameter	Type	Description
username	TT_CHAR (30)	If NULL, all users are returned.

Result set **ttUsers** returns the result set:

Column	Type	Description
username	TT_CHAR(30) NOT NULL	User names of TimesTen instance users.
identification	TT_INTEGER NOT NULL	0 - The user is an internal TimesTen user. 1 - User is identified externally.
privilege_map	BINARY (16) NOT NULL	A binary representation of the user privileges primarily intended for TimesTen internal use. See **ttUserPrivileges** for a human-readable representation of user privileges.

Examples To generate a list of all TimesTen instance users, use:

```
Command> CALL ttUsers;
< USER1, 1, FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF >
< SYS, 0, 00000000000000000000000000000000 >
< TTREP, 0, 00000000000000000000000000000000 >
< PUBLIC, 0, 03000000000000000000000000000000 >
< USER2, 0, 00010000000000000000000000000000 >
< USER3, 1, 00000000000000000000000000000000 >
```

To generate information on USER2, use:

```
Command> CALL ttUsers('user2');
< USER2, 0, 00010000000000000000000000000000 >
```

ttWarnOnLowMemory

Description Allows applications to specify that operations executed on the current connection should return a warning if they allocate memory and find that memory is low. If the value is set, a warning is returned for any operation that does an allocation and finds total memory in use to be above the connection's threshold value as specified by the **PermWarnThreshold** and **TempWarnThreshold** data store attributes. See the "TimesTen Data Store Attributes" Chapter for more information.

Syntax

```
ttWarnOnLowMemory(permanent, temporary)
```

Parameters **ttWarnOnLowMemory** has these parameters:

Parameter	Type	Description
permanent	TT_INTEGER NOT NULL	1(enable) or 0 (disable) warnings for the permanent data partition.
temporary	TT_INTEGER NOT NULL	1 (enable) or 0 (disable) warnings for the temporary data partition.

Result Set **ttWarnOnLowMemory** returns no results.

Example

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

Enables low memory warnings for the permanent data partition only.

Notes By default, low memory warnings are not issued for either partition. Applications that want to receive these warnings must call this procedure. This procedure is connection specific, and so needs to be issued for each connection upon which warnings are desired. Also, the current setting does not persist to subsequent connections.

ttXactIdGet

Description

Displays transaction ID information for interpreting lock messages. The two result columns of **ttXactIdGet** are used in combination to uniquely identify a transaction in a data store. 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.

Syntax

```
ttXactIdGet
```

Parameters

ttXactIdGet has no parameters.

Result set

ttXactIdGet returns the result set:

Column	Type	Description
xactID	TT_INTEGER	Connection ID.
counter	TT_BIGINT	An increasing number that is used to discriminate successive transactions of the same Transaction Id.

Example

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

Note

The output correlates to the values printed in lock error messages and **ttXactAdmin** lock information output.

See Also

"ttXactAdmin" on page 186
ttXactIdRollback in the *Oracle TimesTen In-Memory Database C Developer's and Reference Guide*

ttXlaBookmarkCreate

Description Creates the specified bookmark.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttXlaBookmarkCreate (bookmark)
```

Parameters **ttXlaBookmarkCreate** has the parameter:

Column	Type	Description
bookmark	VARCHAR (31) NOT NULL	The name of the bookmark to be created.

Result set **ttXlaBookmarkCreate** returns no results.

Example

```
Command > call ttXlaBookmarkCreate(mybookmark);
```

See Also

"ttXlaSubscribe" on page 333
"ttXlaUnsubscribe" on page 334
"ttXlaBookmarkDelete" on page 332

ttXlaBookmarkDelete

Description

Deletes the specified bookmark. The bookmark cannot be deleted while it is in use.

Access Control

If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttXlaBookmarkDelete (bookmark)
```

Parameters

ttXlaBookmarkDelete has the parameter:

Column	Type	Description
bookmark	TT_CHAR (31) NOT NULL	The name of the bookmark to be deleted.

Result set

ttXlaBookmarkDelete returns no results.

Example

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

See Also

"ttXlaBookmarkCreate" on page 331
"ttXlaSubscribe" on page 333
"ttXlaUnsubscribe" on page 334

ttXlaSubscribe

Description Sets up persistent XLA tracking of a table. This procedure cannot be executed when the specified bookmark is in use.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

Syntax

```
ttXlaSubscribe (tblName, bookmark)
```

Parameters **ttXlaSubscribe** has the parameters:

Column	Type	Description
tblName	TT_CHAR (61) NOT NULL	The name of the table to be tracked.
bookmark	TT_CHAR (61) NOT NULL	The name of the bookmark that the application uses to track this table.

Result set **ttXlaSubscribe** returns no results.

Example

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

See Also

"ttXlaBookmarkCreate" on page 331
"ttXlaBookmarkDelete" on page 332
"ttXlaUnsubscribe" on page 334

ttXlaUnsubscribe

Description Stops persistent XLA tracking of a table. This procedure cannot be executed when the specified bookmark is in use.

Access Control If Access Control is enabled for your instance of TimesTen, this procedure requires ADMIN privileges.

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.
bookmark	TT_CHAR (61) NOT NULL	The name of the bookmark that the application uses to track this table.

Result set **ttXlaSubscribe** returns no results.

Example

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

See Also

"ttXlaBookmarkCreate" on page 331
"ttXlaBookmarkDelete" on page 332
"ttXlaSubscribe" on page 333

4

System Limits

This chapter lists all TimesTen system limits and defaults. Specific operating system limits may take precedence over these values. Please also see "Installation prerequisites" in the *Oracle TimesTen In-Memory Database Installation Guide* for more details.

Description	32-bit Value	64-bit Value
Maximum number of replication subscriber data stores. (For Active/Standby schemes, one subscriber value is used by the system.)	128	128
Minimum data store size (bytes). Size includes both the permanent and temporary space required to perform operations on the data store.	5MB = 5,242,880	5MB = 5,242,880
Maximum length for a fixed-length column (bytes).	8,300	8,300
Maximum number of columns in a table.	1,000	1,000
Maximum number of columns in an ORDER BY clause.	1,000	1,000
Maximum number of columns in an GROUP BY clause.	1,000	1,000
Maximum inline row width	32,768	32,768
Maximum cumulative length of a row's fixed-length columns (bytes).	8,400	8,400
Maximum number of rows in a table.	256 M = 268,435,256	$(2^{31}-1)$ = 2,147,483,647
Maximum length for a varying-length column (bytes).	2^{22} = 4,194,304	2^{22} = 4,194,304

Description	32-bit Value	64-bit Value
Maximum number of concurrent connections to data stores per instance.[a]	2047	2047
Maximum length of data store names.	32	32
Maximum length of the path name for a data store in an asynchronous writethrough cache group	248	248
Maximum number of projected expressions in a SELECT statement.	32,767	32,767
Maximum length of string specifying a join order.	1,024	1,024
Maximum number of columns in an index (or primary) key.	16	16
Maximum length of basic names.	30	30
Maximum length of displayed predicate string in the PLAN table.	1,024	1,024
Maximum length of SQL statement, including the NULL terminator.	409,600	409,600
Maximum number of table references in an SQL query.	24	24
Maximum number of indexes on a table.	32	32
Maximum number of partitions in a table	255	255
Maximum number of concurrent connections to a TimesTen Server.	9,999	9,999
Maximum number of concurrent shared memory segment client/server connections	512	512
Maximum number of allocated statement handles per shared memory segment client/server connection.	512	512

Description	32-bit Value	64-bit Value
Maximum depth of nesting subqueries.	Equal to the maximum number of table references in an SQL query.	Equal to the maximum number of table references in an SQL query.
Maximum error message length for applications that specify an error message length, for example through a call to SQLError.	512	512

a. Some instances may support a slightly smaller maximum number of connections depending on such things as whether the data store is shared or replicated and operating system limits. Most configurations support no less than 2,000 connections.

Limits on number of open files

Each process connected to a TimesTen data store 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 data store connection:

- Connections to data stores 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 data store checkpoints issued by the process.
- Additional file descriptors may be opened during transaction commit or rollback operations.

For multithreaded applications that maintain many concurrent TimesTen data store connections, the default number of open files permitted to each process by the operating system may be too low.

- On HP-UX, the default is 4096 open files per process and may be raised through the tunable parameter *maxfiles* or with the `ulimit` command (`limit` for csh users). You can also set the per-process limit programmatically with `setrlimit`.

- 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 run into 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.

- On Tru64, the limit is 2,046 open files, so you are not likely to run into problems.

Most of the open file descriptors are used for reading and writing data store recovery log files. If a process fails to open a log file, the data store is marked as requiring recovery and all current connections to the data store are terminated.

Path names

TimesTen does not support file path names that contain multi-byte characters. Please make sure that the installation path, data store path, transaction log path, and temporary file path do not contain any multibyte characters.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W