Oracle TimesTen In-Memory Database Application Server Configuration Guide

Release 6.0

B25268-03

For the latest updates, refer to the TimesTen release notes.

March 2006

Printed in the United States of America

Contents

About this Guide

TimesTen® is a high-performance, in-memory data manager that supports the ODBC and JDBC interfaces. The examples and procedures in this guide use the JDBC interface.

This guide is for application developers who use and administer TimesTen JDBC and for system administrators who configure and manage the TimesTen Daemon. It provides information about configuring your application server for use with TimesTen.

To work with this guide, you should be familiar with the use of both your application server and TimesTen/DataServer.

TimesTen documentation

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

- The *Oracle TimesTen In-Memory Database Installation Guide* provides information needed to install and configure TimesTen on all supported platforms.
- The *Oracle TimesTen In-Memory Database Architectural Overview* provides a description of all the available features in TimesTen.
- The *Oracle TimesTen In-Memory Database Operations Guide* provides information on configuring TimesTen and using the ttIsql utility to manage a data store. This guide also provides a basic tutorial for TimesTen.
- The *Oracle TimesTen In-Memory Database C Developer's and Reference Guide* and the *Oracle TimesTen In-Memory Database Java Developer's and Reference Guide* provide information on how to use the full set of available features in TimesTen to develop and implement applications that use TimesTen.
- The *Oracle TimesTen In-Memory Database API and SQL Reference Guide* contains a complete reference to all TimesTen utilities, procedures, APIs and other features of TimesTen.
- The *TimesTen to TimesTen Replication Guide*. This guide is for application developers who use and administer TimesTen and for system administrators who configure and manage TimesTen Replication. It provides: Background information to help you understand how TimesTen Replication works. Step-by-step instruction and examples that show how to perform the most commonly needed tasks.
- The *TimesTen Cache Connect to Oracle Guide* describes how to use Oracle Connect to cache Oracle data in TimesTen. This guide is for developers who use and administer TimesTen for caching Oracle data. It provides information

on caching Oracle data in TimesTen data stores. It also describes how to use the Oracle Connect Administrator, a web-based interface for creating cache groups.

TimesTen documentation is available on the product CD-ROM and on the TimesTen web site: `http://www.timesten.com`.

Installing TimesTen

TimesTen Release 6.0 includes the TimesTen Data Manager for 32-bit and 64-bit platforms. See the *Oracle TimesTen In-Memory Database Installation Guide* for a description of supported platforms.

In addition to the Data Manager, TimesTen Release 6.0 also includes TimesTen Client and Server components. You can install the TimesTen Data Manager stand-alone or in a client/server environment.

For a list of the The TimesTen default installation directories, see the *Oracle TimesTen In-Memory Database Installation Guide*.

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`.`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

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

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

If you see...	It means...
install_dir	The path that represents the directory where the current release of TimesTen is installed.
TTinstance	The instance name for your specific installation of TimesTen. Each installation of TimesTen must be identified at install time with a unique alphanumeric instance name. This name appears in the install path. The instance name “giraffe” is used in examples in this guide.
`bits` or *`bb`*	Two digits, either 32 or 64, that represent either the 32-bit or 64-bit operating system.
`release` *or* *`rr`*	Two digits that represent the first two digits of the current TimesTen release number, with or without a dot. For example, 51 or 5.0 represents TimesTen Release 5.0.
jdk_version	Two digits that represent the version number of the major JDK release. For example 14 for versions of jdk1.4.
`timesten`	A sample name for the TimesTen instance administrator. You can use any legal user name as the TimesTen administrator. On Windows, the TimesTen instance administrator must be a member of the Administrators group. Each TimesTen instance can have a unique instance administrator name.
`DSN`	The data source name.

Technical Support

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

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

Email: timesten-support_us@oracle.com

1

Configuring Oracle Application Server and TimesTen

This section provides descriptions and examples of configuring the TimesTen JDBC driver for use with Oracle Application Server 10g. The document assumes that the reader has a basic familiarity with the use of both Oracle Application Server and TimesTen Data Server.

This chapter includes the following topics:

Configuring the TimesTen JDBC Driver Classes

To configure Oracle Application Server for use with the TimesTen JDBC drivers, set the `LD_LIBRARY_PATH` environment variable for the OC4J server process to access the TimesTen shared libraries. Environment variables for the OC4J server process can be set using the web-based Enterprise Manager 10g console.

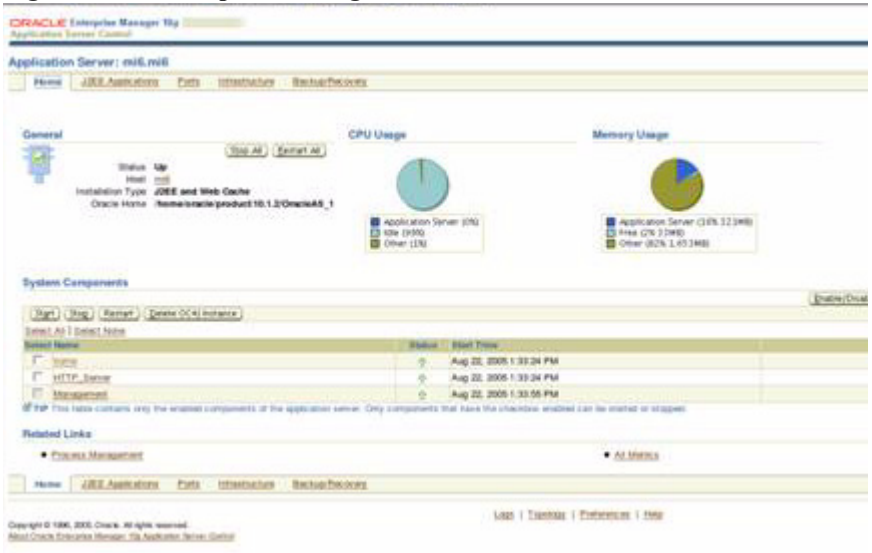
Note: `LD_LIBRARY_PATH` is the environment variable on Linux and Solaris. Set the appropriate path variable for your operating system. For a complete description of the environment variable requirements for a particular platform, please see the "Environment modifications" section of the *Oracle TimesTen In-Memory Database Installation Guide*.

---

1. Using a Web browser, connect to Enterprise Manager. (By default, the URL is `http://`*`servername`*`:1810`.) Enter the user ID (typically `ias_admin`) and password specified during the installation of Oracle Application Server.

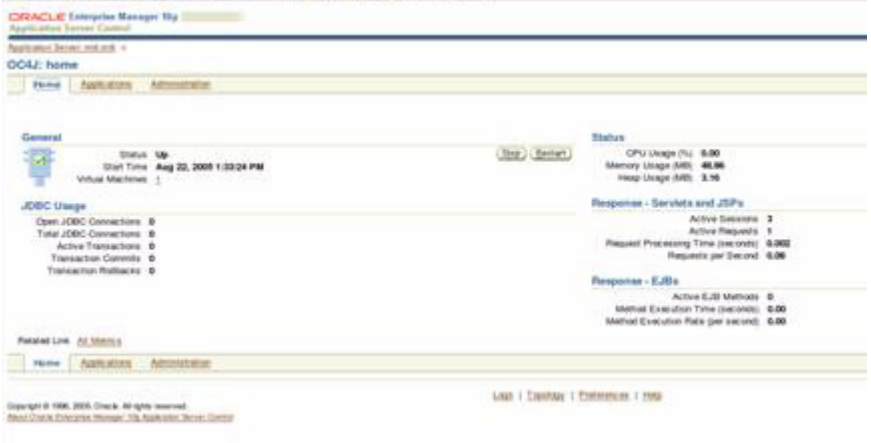
   You should see a screen similar to the one in Figure1.1.

Figure 1.1 Enterprise Manager



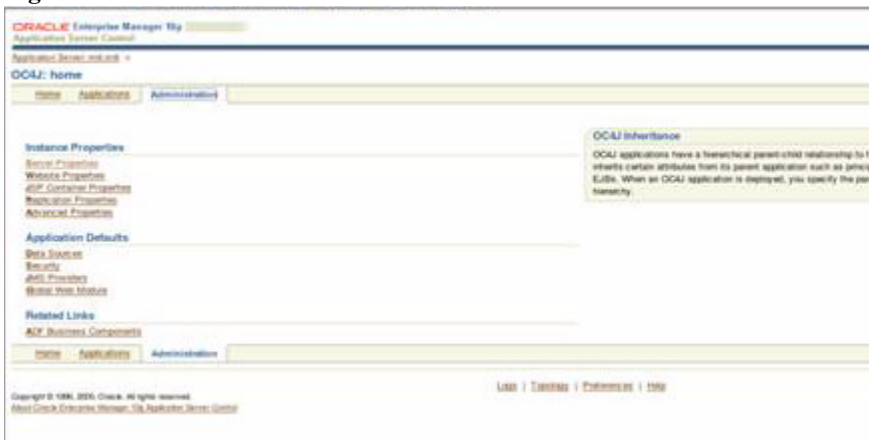
2. In the middle of the screen, select the name of the OC4J container for TimesTen. By default this is **home** (the first choice). A screen similar to the one in Figure 1.2 results.

Figure 1.2 OC4J container



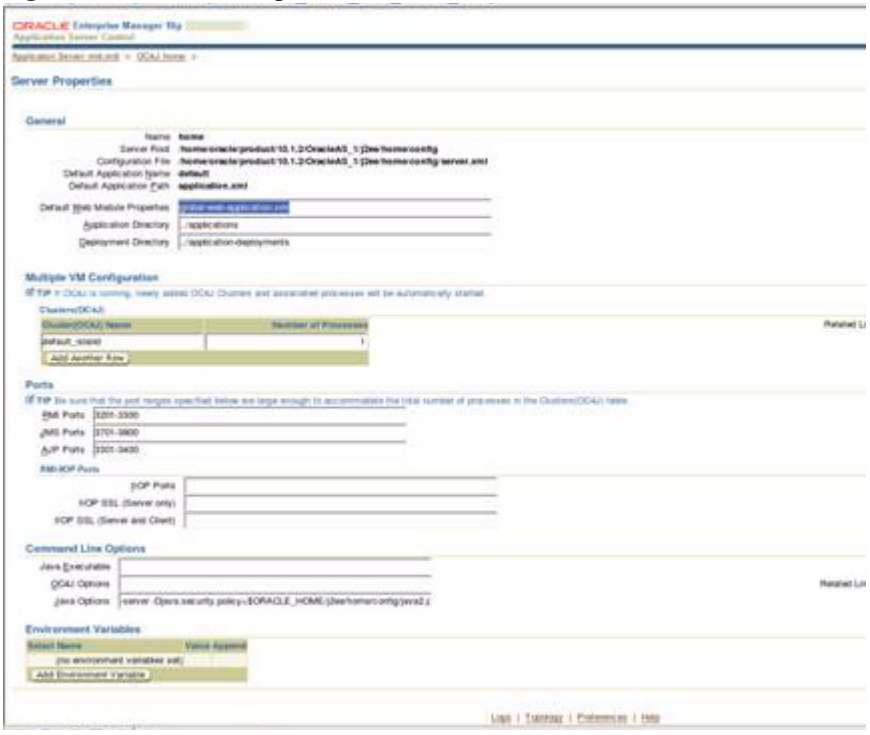
3. Click the **Administration** tab in the upper left. A screen similar to the one in Figure 1.3 results.

Figure 1.3 OC4J Administration



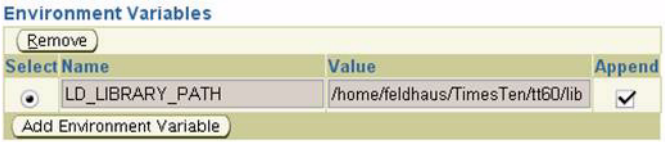
4. Under Instance Properties, click **Server Properties**. A screen similar to the one in Figure 1.4 results.

Figure 1.4 Server Properties



5. The Environment Variables section is at the bottom of the Server Properties screen. Assign the `LD_LIBRARY_PATH` environment variable to the location of the TimesTen `lib` directory. See Figure 1.5.

Figure 1.5 Setting LD_LIBRARY_PATH



6. Assign a symbolic link to the TimesTen JDBC driver JAR file (`classes14.jar`).

   a. Change to the J2EE applications library.

   ```
   cd OAS_install_directory/j2ee/home/applib
   ```

   b. Assign the symbolic link to the TimesTen JDBC driver JAR file.

   ```
   ln -s timesten_install_directory/lib/classes14.jar
   ```

Configuring TimesTen Data Sources

Create TimesTen DSNs in the either the `$HOME/.odbc.ini` or `/var/TimesTen/sys.odbc.ini` file for use by Oracle Application Server. The following DSN definition was used for the examples in this section:

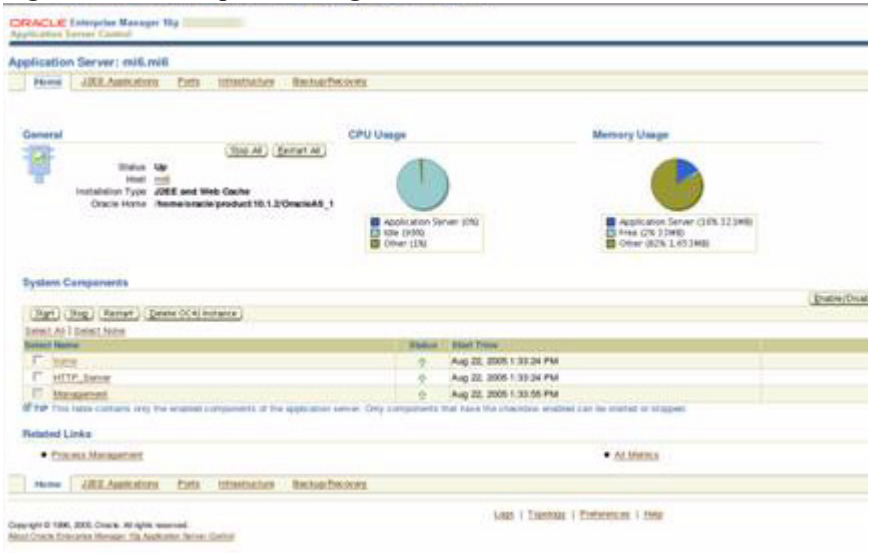
```
[ORACLEAS]
DataStore=/home/oracleas/ds/ORACLEAS
PermSize=64
OracleID=europa
OraclePWD=tiger
SMPOptLevel=1
DurableCommits=0
UID=scott
PWD=tiger
```

In order for the Oracle Application Server applications to access TimesTen, Data Sources must be configured for TimesTen in the OC4J server. You can use Enterprise Manager to do this.

1. Using a Web browser, connect to Enterprise Manager. (By default, the URL is `http://servername:1810`.) Enter the user ID (typically `ias_admin`) and password specified during the installation of Oracle Application Server.

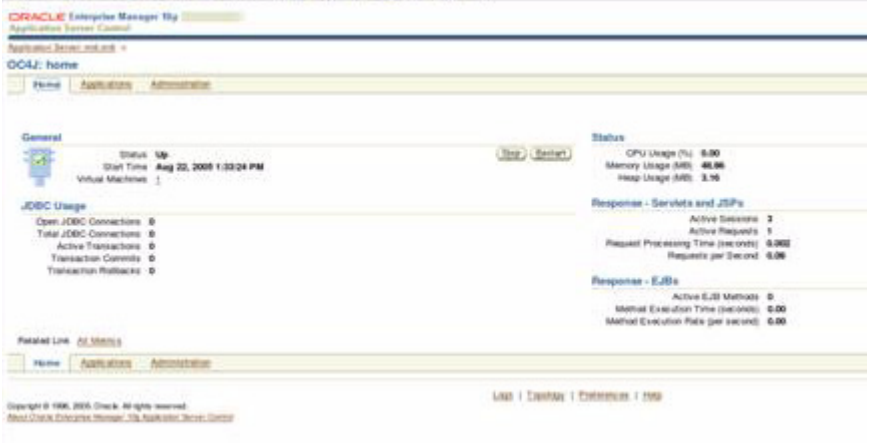
   You should see a screen similar to the one in Figure1.1.

Figure 1.6 Enterprise Manager



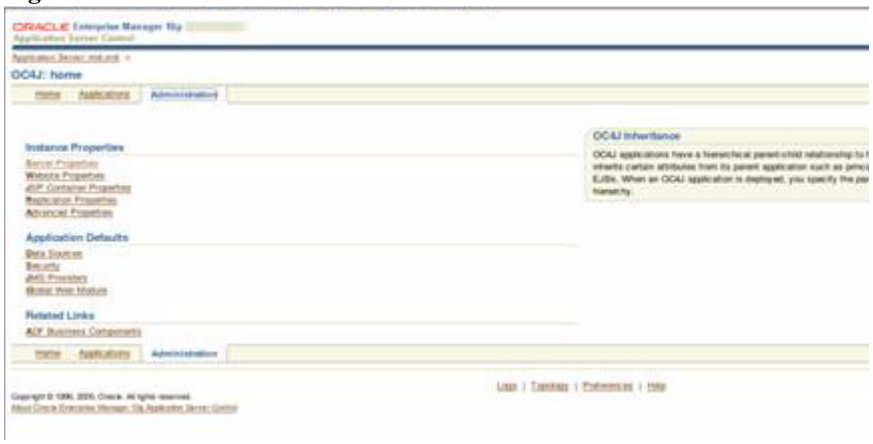
2. In the middle of the screen, select the name of the OC4J container for TimesTen. By default this is **home** (the first choice). A screen similar to the one in Figure 1.2 results.

Figure 1.7 OC4J container



3. Click the **Administration** tab in the upper left. A screen similar to the one in Figure 1.3 results.

Figure 1.8 OC4J Administration

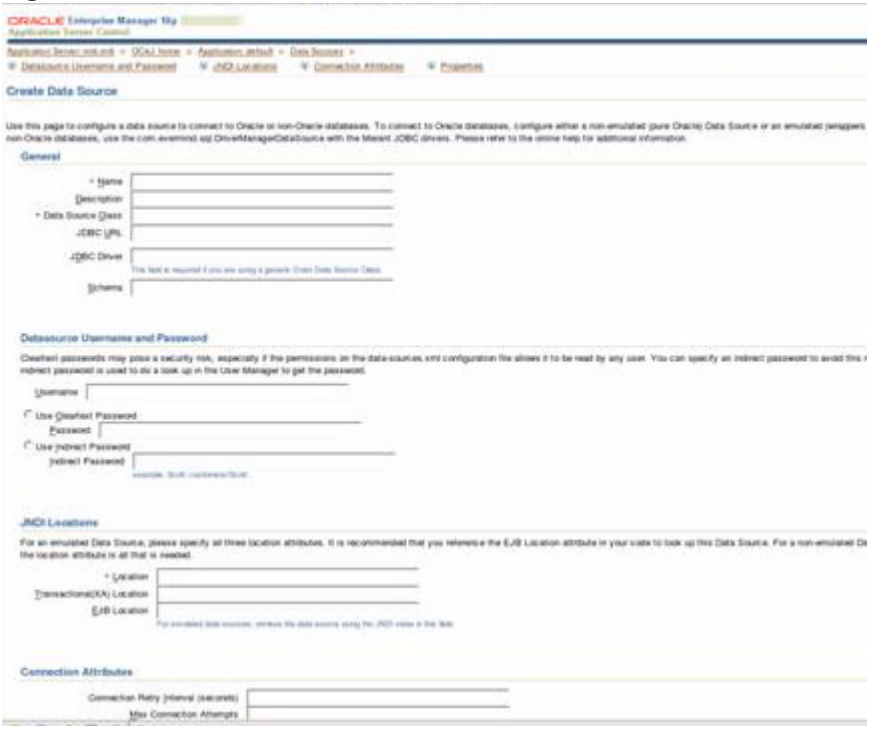


4. Under Application Defaults, click **Data Sources**. A screen similar to the one in Figure 1.9 results

Figure 1.9 Application Defaults

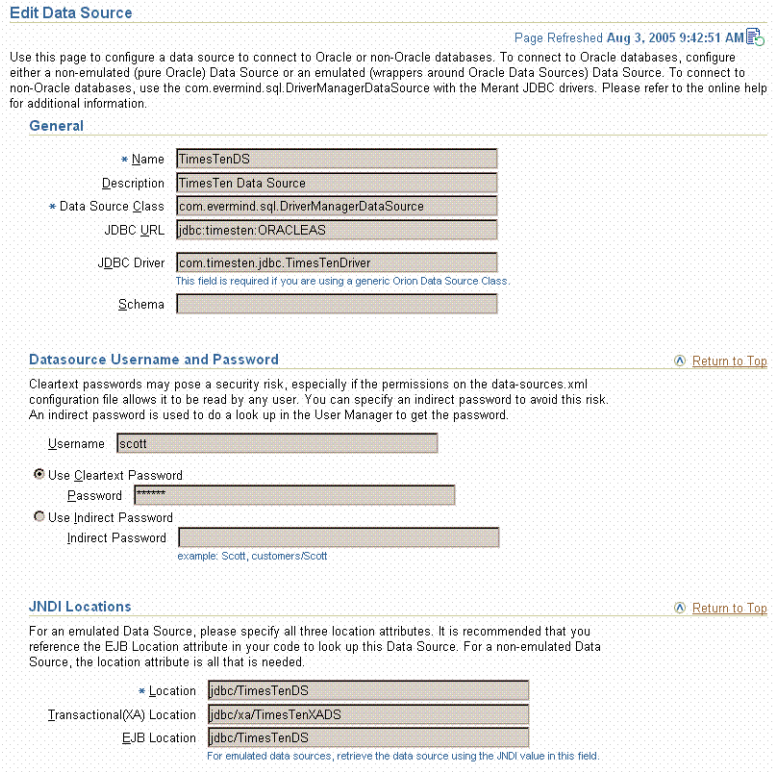
5. To create a new data source for TimesTen, click Create (on the right side of the screen). A screen similar to the one in Figure 1.10 results.

Figure 1.10 Create Data Source



6. Configure the TimesTen data source in the OC4J server, as shown in Figure 1.11.

Figure 1.11 Configuring TimesTen Data Sources in the OC4J Server



Data Source configuration changes made in the console are written to a file called `config/data-sources.xml` in the OC4J server's directory.

Oracle Application Server defines three different types of Data Source configurations. Only emulated data sources are recommended when using TimesTen. Emulated Data Sources are data sources that emulate the XA protocol for JTA transactions. However, they do not support the two-phase XA commit protocol. A sample configuration for an emulated TimesTen Data Source is presented below as it appears in the OC4J server `data-sources.xml` file. (If you prefer to use the Enterprise Manager 10g console GUI to configure the TimesTen emulated Data Source, you can use the following XML descriptor as a guide to the GUI configuration.)

```
<data-source
location="jdbc/TimesTenDS"
class="com.evermind.sql.DriverManagerDataSource"
xa-location="jdbc/xa/TimesTenXADS"
ejb-location="jdbc/TimesTenDS"
connection-driver="com.timesten.jdbc.TimesTenDriver"
username="scott"
password="tiger"
```

```
min-connections="8"
max-connections="64"
url="jdbc:timesten:ORACLEAS"
name="TimesTenDS">

<description>TimesTen Data Source</description>
</data-source>
```

This emulated Data Source configuration allows TimesTen to be used with entity beans and other components requiring JTA transaction support. It also uses the OC4J connection pool service.

2

Configuring JBoss and TimesTen

This section describes how to configure the JBoss 4 Application Server to use the TimesTen JDBC driver. It includes the following topics:

Configuring the TimesTen JDBC Driver Classes

To configure JBoss for use with the TimesTen JDBC driver classes perform the steps:

1. In the shell where the JBoss server is started, make sure that the `LD_LIBRARY_PATH` environment variable includes the path to the lib directory of the TimesTen installation. For example:

   ```
   [jboss@europa bin]$ export LD_LIBRARY_PATH=/home/timesten/TimesTen/
       tt60/lib
   ```

   Note: For a complete description of the environment variable requirements for a particular platform, please see the "Environment modifications" section of the *Oracle TimesTen In-Memory Database Installation Guide*.

2. Create a symbolic link from the `server/default/lib` directory of the JBoss installation to the TimesTen JDBC driver jar file. For example:

   ```
   [jboss@europa lib]$ pwd
   /home/jboss/jboss-4.0.1/server/default/lib
   [jboss@europa lib]$ ln -s /home/timesten/TimesTen/tt60/jdbc/lib/
   classes14.jar
   ```

   The JBoss server can now be started with the ability to access the TimesTen JDBC driver classes.

Configuring the TimesTen Type Mapping

For TimesTen to work with various features of the JBoss server, a `type-mapping` XML descriptor must be configured to tell JBoss how to work with the TimesTen SQL dialect. See section 11.13 of the *JBoss 4 Application Server Guide* for a complete discussion of this configuration.

A `type-mapping` descriptor is normally included in the `standardjbosscmp-jdbc.xml` file located in the `conf` directory of the JBoss server. A sample `type-mapping` descriptor for TimesTen is shown in Example 2.1.

Example 2.1

```
<type-mapping>

      <name>TimesTen</name>
      <row-locking-template>
         SELECT ?1 FROM ?2 WHERE ?3 FOR UPDATE
      </row-locking-template>
      <pk-constraint-template>
         PRIMARY KEY (?2)
      </pk-constraint-template>
      <fk-constraint-template>
         ALTER TABLE ?1 ADD CONSTRAINT ?2 FOREIGN KEY (?3) REFERENCES ?4 (?5)
      </fk-constraint-template>
      <add-column-template>
         ALTER TABLE ?1 ADD COLUMN ?2 ?3
      </add-column-template>
      <drop-column-template>
         ALTER TABLE ?1 DROP COLUMN ?2
      </drop-column-template>

      <alias-header-prefix>t</alias-header-prefix>
      <alias-header-suffix>_</alias-header-suffix>
      <alias-max-length>30</alias-max-length>
      <subquery-supported>true</subquery-supported>
      <true-mapping>(1=1)</true-mapping>
      <false-mapping>(1=0)</false-mapping>

      <function-mapping>
         <function-name>concat</function-name>
         <function-sql>CONCAT (?1, ?2)</function-sql>
      </function-mapping>
      <function-mapping>
         <function-name>substring</function-name>
         <function-sql>SUBSTRING (?1, ?2, ?3)</function-sql>
      </function-mapping>
      <function-mapping>
         <function-name>lcase</function-name>
         <function-sql>LOWER (?1)</function-sql>
      </function-mapping>
```

```
<function-mapping>
   <function-name>length</function-name>
   <function-sql>LENGTH (?1)</function-sql>
</function-mapping>
<function-mapping>
   <function-name>locate</function-name>
   <function-sql>INSTR (?2, ?1, ?3)</function-sql>
</function-mapping>
<function-mapping>
   <function-name>ucase</function-name>
   <function-sql>UPPER (?1)</function-sql>
</function-mapping>
<function-mapping>
   <function-name>count</function-name>
   <function-sql>COUNT (?1)</function-sql>
</function-mapping>

<mapping>
   <java-type>java.lang.Boolean</java-type>
   <jdbc-type>TINYINT</jdbc-type>
   <sql-type>TINYINT</sql-type>
</mapping>
<mapping>
   <java-type>java.lang.Byte</java-type>
   <jdbc-type>TINYINT</jdbc-type>
   <sql-type>TINYINT</sql-type>
</mapping>
<mapping>
   <java-type>java.lang.Short</java-type>
   <jdbc-type>SMALLINT</jdbc-type>
   <sql-type>SMALLINT</sql-type>
</mapping>
<mapping>
   <java-type>java.lang.Integer</java-type>
   <jdbc-type>INTEGER</jdbc-type>
   <sql-type>INTEGER</sql-type>
</mapping>
<mapping>
   <java-type>java.lang.Long</java-type>
   <jdbc-type>BIGINT</jdbc-type>
   <sql-type>BIGINT</sql-type>
</mapping>
<mapping>
   <java-type>java.lang.Float</java-type>
   <jdbc-type>REAL</jdbc-type>
   <sql-type>REAL</sql-type>
</mapping>
<mapping>
   <java-type>java.math.BigDecimal</java-type>
```

```
            <jdbc-type>DECIMAL</jdbc-type>
            <sql-type>DECIMAL (40,15)</sql-type>
        </mapping>
        <mapping>
            <java-type>java.lang.Double</java-type>
            <jdbc-type>DOUBLE</jdbc-type>
            <sql-type>DOUBLE</sql-type>
        </mapping>
        <mapping>
            <java-type>java.lang.Character</java-type>
            <jdbc-type>CHAR</jdbc-type>
            <sql-type>CHARACTER</sql-type>
        </mapping>
        <mapping>
            <java-type>java.lang.String</java-type>
            <jdbc-type>VARCHAR</jdbc-type>
            <sql-type>VARCHAR (256)</sql-type>
        </mapping>
        <mapping>
            <java-type>java.sql.Date</java-type>
            <jdbc-type>DATE</jdbc-type>
            <sql-type>DATE</sql-type>
        </mapping>
        <mapping>
            <java-type>java.sql.Time</java-type>
            <jdbc-type>TIME</jdbc-type>
            <sql-type>TIME</sql-type>
        </mapping>
        <mapping>
            <java-type>java.sql.Timestamp</java-type>
            <jdbc-type>TIMESTAMP</jdbc-type>
            <sql-type>TIMESTAMP</sql-type>
        </mapping>
        <mapping>
            <java-type>java.lang.Object</java-type>
            <!-- directly serializable objects -->
            <jdbc-type>VARBINARY</jdbc-type>
            <sql-type>VARBINARY (4194304)</sql-type>
        </mapping>
</type-mapping>
```

To configure JBoss for this TimesTen type mapping, copy the descriptor into the `type-mappings` section of the `conf/standardjbosscmp-jdbc.xml` configuration file in your JBoss server directory.

Not all elements of the `type-mappings` descriptor are supported by the TimesTen SQL dialect as noted below:

- The `auto-increment-template` element is not supported. Current versions of TimesTen do not support auto increment columns, although SEQUENCES are supported in TimesTen
- The ABS and SQRT functions are not supported in TimesTen.

The Java type to SQL type mappings in the sample `type-mapping` descriptor for TimesTen can be modified to improve performance based on application requirements. For example, the sample configuration maps a `java.lang.String` to a TimesTen SQL type of `VARCHAR (256)`. Performance would improve if this were changed to a TimesTen SQL type of `CHAR (16)`.

Configuring TimesTen Data Sources

Create TimesTen DSNs in the `$HOME/.odbc.ini` or `sys.odbc.ini` files for use by the JBoss server. The examples in this guide use the DSN configurations shown in example 2.2:

Example 2.2

```
[JBOSS]
DataStore=/home/jboss/ds/JBOSS
PermSize=64
UID=jboss
PWD=jboss

[JBOSS_CS]
TTC_SERVER=LocalHost_tt60
TTC_SERVER_DSN=JBOSS
UID=jboss
PWD=jboss

[JBOSS_JMS]
DataStore=/home/jboss/ds/JBOSS_JMS
PermSize=64
UID=jboss
PWD=jboss
```

For JBoss applications to access TimesTen data source connection pools, a deployment descriptor with the name pattern of `*-ds.xml` must be created in the `deploy` directory of the JBoss server. See section 7.3 of the of the *JBoss 4 Application Server Guide* for a complete discussion of this configuration.

A sample configuration file called `timesten-ds.xml` is included in Appendix A. To use this configuration, copy the file as `timesten-ds.xml` to the `deploy` directory of the JBoss server.

This data source configuration file defines four distinct types of connections that can be configured for TimesTen.

- The `TimesTenLocalDS` configuration is for a non-XA direct connection to a TimesTen data source located on the same machine as the JBoss server.
- The `TimesTenLocalClientDS` configuration is for a non-XA client/server connection to a TimesTen data source that can be located on a different machine than the JBoss server.
- The `TimesTenXADS` configuration is for an XA direct connection to a TimesTen data source located on the same machine as the JBoss server.
- The `TimesTenXAClientDS` configuration is for a XA client/server connection to a TimesTen data source that can be located on a different machine than the JBoss server.
- The `TimesTenJMS` configuration is identical to the `TimesTenXADS` configuration. This data source is used for storing messages associated with durable JMS subscribers.

Using TimesTen with JBoss JMS

The JBoss implementation of the JMS service uses an RDBMS to provide support for durable topic subscriptions and the users associated with those subscriptions.

TimesTen can be configured as the RDBMS for JBoss JMS. Two XML files included in Appendix A, `timesten-jdbc2-service.xml` and `timesten-jdbc-state-service.xml`, provide a configuration that is compatible with a TimesTen data source. Both files reference a data source called `TimesTenJMS`. This data source was defined in the `TimesTenJMS` configuration described in the previous section.

To enable this configuration copy the `timesten-jdbc-2-service.xml` and the `timesten-jdbc-state-service.xml` files to the `deploy/jms` directory of the JBoss server. Be sure to remove any previous configuration files for the JMS RDBMS in the directory. When the JBoss server is started the necessary JMS message and state tables will be automatically created in the TimesTenJMS data store.

The EJB Timer Service

The EJB Timer Service is a container service that allows EJBs to register for callbacks by the container at specified intervals. EJB Timers can be persisted to a data source in JBoss. To use a TimesTen data source for EJB Timer persistence do the following:.

1. Create this table in the TimesTen data source:

```
CREATE TABLE TIMERS (
TIMERID VARCHAR (80) NOT NULL,
TARGETID VARCHAR (80) NOT NULL,
```

```
INITIALDATE TIMESTAMP NOT NULL,
TIMERINTERVAL DECIMAL (20),
INSTANCEPK VARBINARY (4194304),
INFO VARBINARY (4194304),
PRIMARY KEY (TIMERID, TARGETID))
```

2. Edit the `deploy/ejb-deployer.xml` configuration file in the JBoss server directory and set the DataSource name attribute for the `org.jboss.ejb.txtimer.DatabasePersistencePolicy` Mbean to use a TimesTen data source. An example using a data source called `TimesTenLocalDS` is shown in Example 2.3.

Example 2.3

```
<!-- A persistence policy that persistes timers to a database -->
<mbean code="org.jboss.ejb.txtimer.DatabasePersistencePolicy"
       name="jboss.ejb:service=EJBTimerService,persistencePolicy=database">

       <!-- DataSource JNDI name -->
       <depends optional-attributename="DataSource">
          jboss.jca:service=DataSourceBinding,name=TimesTenLocalDS
       </depends>
       <!-- The plugin that handles database persistence -->
       <attribute name="DatabasePersistencePlugin">
          org.jboss.ejb.txtimer.GeneralPurposeDatabasePersistencePlugin
       </attribute>
</mbean>
```

3

Configuring Weblogic and TimesTen

This section describes how to configure Weblogic Application Server 8 to use the TimesTen JDBC driver. It includes the following topics:

- Configuring the TimesTen JDBC Driver Classes
- Configuring TimesTen Connection Pools and Data Sources

Configuring the TimesTen JDBC Driver Classes

To configure Weblogic for use with the TimesTen JDBC drivers the environment where the Weblogic Server is started must be modified to access both the JDBC driver classes and the TimesTen shared libraries. To set the `LD_LIBRARY_PATH` and `CLASSPATH` environment variables, run the script `ttSetEnv`. The Weblogic server can now be started with the ability to access TimesTen data stores.

Note: For a complete description of the environment variable requirements for a particular platform, please see the "Environment modifications" section of the *Oracle TimesTen In-Memory Database Installation Guide*.

Configuring TimesTen Connection Pools and Data Sources

Create TimesTen DSNs in the `$HOME/.odbc.ini` or `sys.odbc.ini` files for use by the Weblogic server. For the examples in this document the following DSNs were used:

```
[WLS]
DataStore=/home/wls/ds/WLS
PermSize=64
UID=wls
PWD=wls

[WLS_CS]
TTC_SERVER=LocalHost_tt60
TTC_SERVER_DSN=WLS
UID=wls
PWD=wls
```

For Weblogic applications to access TimesTen data stores, both connection pools and associated DataSources must be configured for the server. A common way to

do this is to use the Weblogic Server console GUI. Configuration changes made in the console are written to a file called `config.xml` in the server's directory.

The configuration information below is presented as it appears in the server's `config.xml` file. If you prefer to use the WebLogic console GUI to configure TimesTen, you can use the XML descriptors below as a guide to the GUI configuration.

There are four distinct types of TimesTen connections that can be configured for Weblogic Server.

- The `TimesTenLocalPool` and `TimesTenLocalDS` configuration shown below is for a non-XA direct connection to a TimesTen data store located on the same machine as the Weblogic server.

```
<JDBCConnectionPool ConnLeakProfilingEnabled="true"
    ConnProfilingEnabled="true"
    DriverName="com.timesten.jdbc.TimesTenDriver"
    InitialCapacity="1"
    MaxCapacity="16"
    Name="TimesTenLocalPool"
    PasswordEncrypted=""
    Properties=""
    StatementCacheSize="32"
    Targets="myserver"
    TestConnectionsOnCreate="true"
    TestConnectionsOnRelease="true"
    TestConnectionsOnReserve="false"
    TestTableName="SYS.MONITOR"
    URL="jdbc:timesten:WLS;UID=wls;PWD=wls"/>

<JDBCTxDataSource EnableTwoPhaseCommit="true"
    JNDIName="TimesTenLocalDS"
    Name="TimesTenLocalDS"
    PoolName="TimesTenLocalPool"
    RowPrefetchEnabled="true"
    Targets="myserver"/>
```

- The `TimesTenLocalClientPool and TimesTenLocalClientDS` configuration is for a non-XA client/server connection to a TimesTen data store that can be located on a different machine than the Weblogic server.

```
<JDBCConnectionPool ConnLeakProfilingEnabled="true"
    ConnProfilingEnabled="true"
    DriverName="com.timesten.jdbc.TimesTenDriver"
    MaxCapacity="16"
    Name="TimesTenLocalClientPool"
    PasswordEncrypted=""
    Properties="user="
    StatementCacheSize="32"
    Targets="myserver"
```

```
    TestConnectionsOnCreate="true"
    TestConnectionsOnRelease="true"
    TestTableName="SYS.MONITOR"
    URL="jdbc:timesten:client:WLS_CS;UID=wls;PWD=wls"/>

<JDBCTxDataSource EnableTwoPhaseCommit="true"
    JNDIName="TimesTenLocalClientDS"
    Name="TimesTenLocalClientDS"
    PoolName="TimesTenLocalClientPool"
    RowPrefetchEnabled="true"
    Targets="myserver"/>
```

- The `TimesTenXAPool` and `TimesTenXADS` configuration is for a XA direct connection to a TimesTen data store located on the same machine as the Weblogic server.

```
<JDBCConnectionPool ConnLeakProfilingEnabled="true"
    ConnProfilingEnabled="true"
    DriverName="com.timesten.jdbc.xa.TimesTenXADataSource"
    KeepLogicalConnOpenOnRelease="true" MaxCapacity="16"
    Name="TimesTenXAPool" NewXAConnForCommit="true"
    PasswordEncrypted="{3DES}y+DgSei/wXo="
    Properties="URL=jdbc:timesten:WLS;user=wls"
    StatementCacheSize="32" SupportsLocalTransaction="true"
    Targets="myserver" TestConnectionsOnCreate="true"
    TestConnectionsOnRelease="true"
    TestTableName="SYS.TABLES"
    URL="jdbc:timesten:WLS"/>
<JDBCTxDataSource
    JNDIName="TimesTenXADS"
    Name="TimesTenXADS"
    PoolName="TimesTenXAPool" R
    owPrefetchEnabled="true"
    Targets="myserver"/>
```

- The `TimesTenXAClientPool` and `TimesTenXAClientDS` configuration is for a XA client/server connection to a TimesTen data store that can be located on a different machine than the Weblogic server.

```
<JDBCConnectionPool ConnLeakProfilingEnabled="true"
    ConnProfilingEnabled="true"
    DriverName="com.timesten.jdbc.xa.TimesTenXADataSource"
    KeepLogicalConnOpenOnRelease="true" MaxCapacity="16"
    Name="TimesTenXAClientPool" NewXAConnForCommit="true"
    PasswordEncrypted="{3DES}y+DgSei/wXo="
    Properties="URL=jdbc:timesten:client:WLS_CS;user=wls"
    StatementCacheSize="32" SupportsLocalTransaction="true"
    Targets="myserver" TestConnectionsOnCreate="true"
    TestConnectionsOnRelease="true"
    TestTableName="SYS.TABLES"
    URL="jdbc:timesten:client:WLS_CS"/>
```

```
<JDBCTxDataSource JNDIName="TimesTenXAClientDS"
    Name="TimesTenXAClientDS" PoolName="TimesTenXAClientPool"
    RowPrefetchEnabled="true" Targets="myserver"/>
```

4

Configuring Sun Java System Application Server and TimesTen

This chapter describes how to configure the Sun Java System Application Server to use the TimesTen JDBC driver. It includes the following topics:

Configuring the JVM Classpath Settings

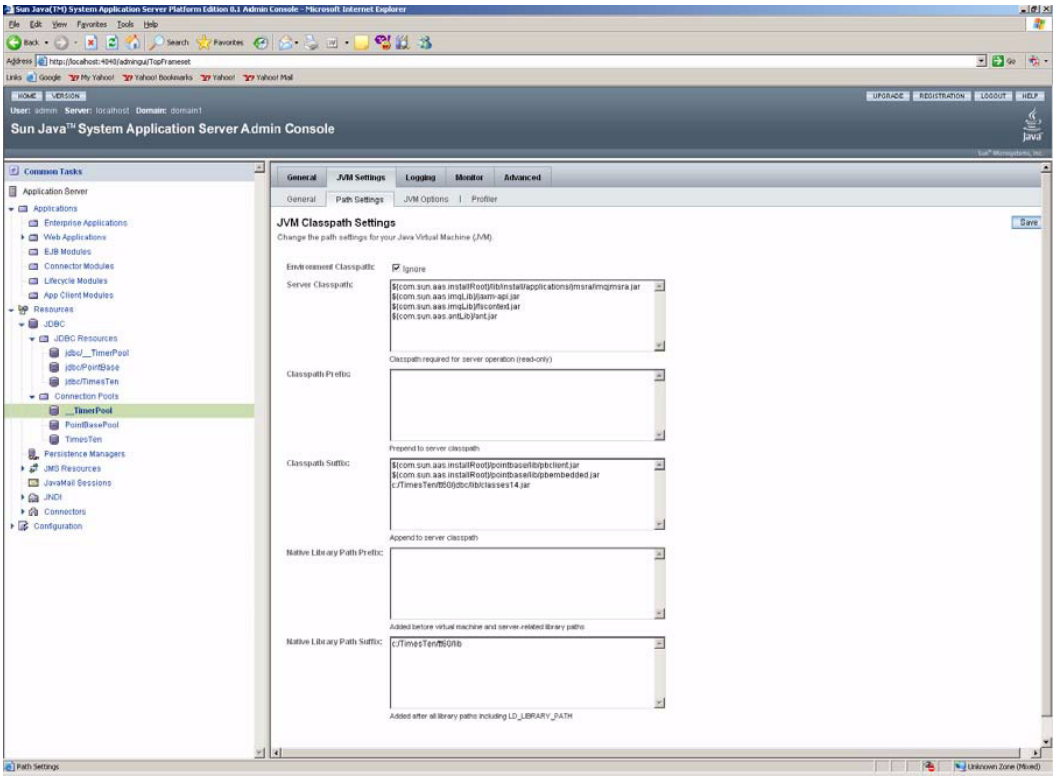
1. Log into the Sun Java ™ System Application Server Admin Console.
2. Navigate to **Application Server -> JVM Settings -> Path Settings**.

Note: For a complete description of the environment variable requirements for a particular platform, please see the "Environment modifications" section of the *Oracle TimesTen In-Memory Database Installation Guide*.

---

3. Add *install_dir*`/jdbc/lib/classes14.jar` to the JVM Classpath Suffix, as shown in Figure 4.1 on page 28.
4. Add *install_dir*`/lib` to the JVM Native Library Path Suffix, as shown in Figure 4.1 on page 28.

Figure 4.1 JVM Classpath Settings

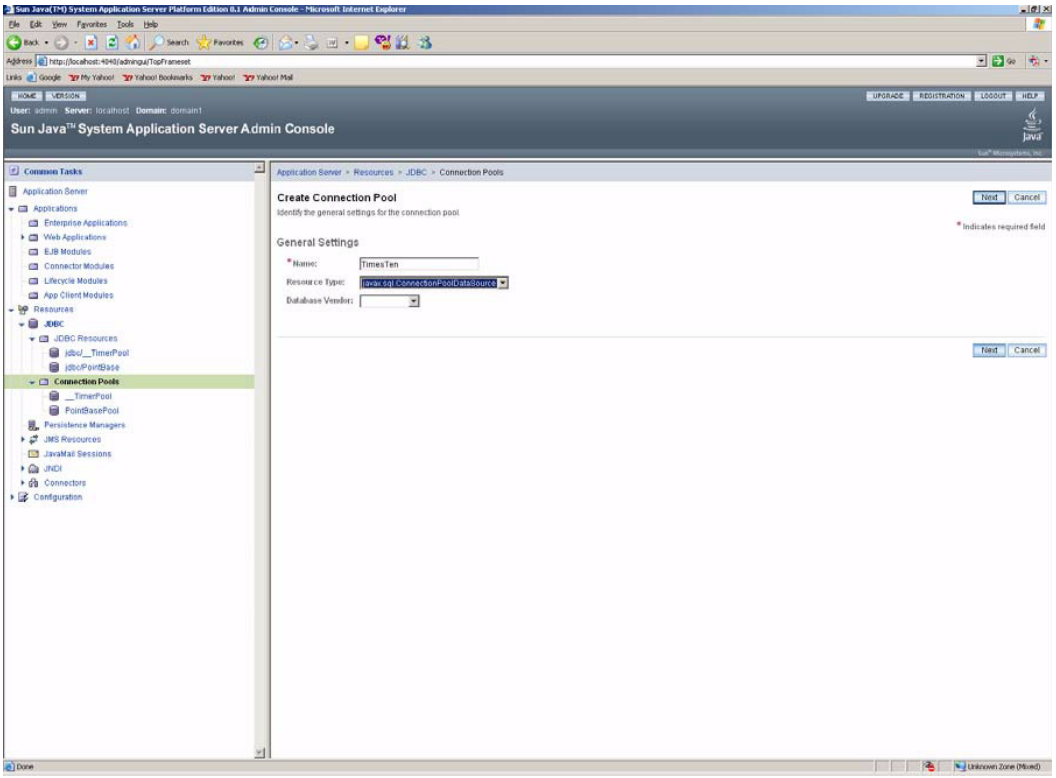


5. Save the changes. A restart may be required at this point.

Add a Connection Pool

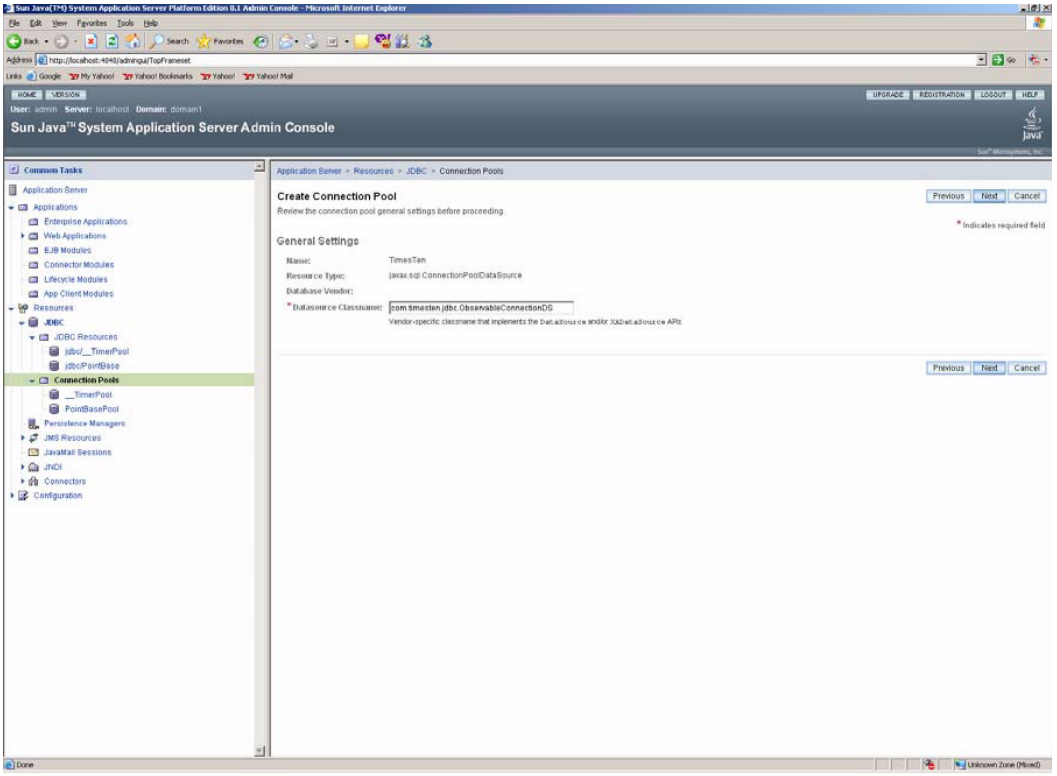
1. Log into the Sun Java System Application Server Admin Console.
2. Navigate to **Resources -> JDBC -> Connection Pools**.
3. Click **New**.
4. To add a pool called TimesTen, specify the following General Settings, as shown in Figure 4.2 on page 29.
   - Name: `TimesTen`.
   - Resource Type: `javax.sql.ConnectionPoolDataSource`.

Figure 4.2 Create Connection Pool



5. Click **Next**.
6. Specify `com.timesten.jdbc.ObservableConnectionDS` as the DataSource ClassName, as shown in Figure 4.3 on page 30.

Figure 4.3 DataSource ClassName



7. Click **Next**.
8. Add Additional Properties as needed. For example, to add a connection pool for the DSN `ttjdbc`:

   a. Click **Add Property**.

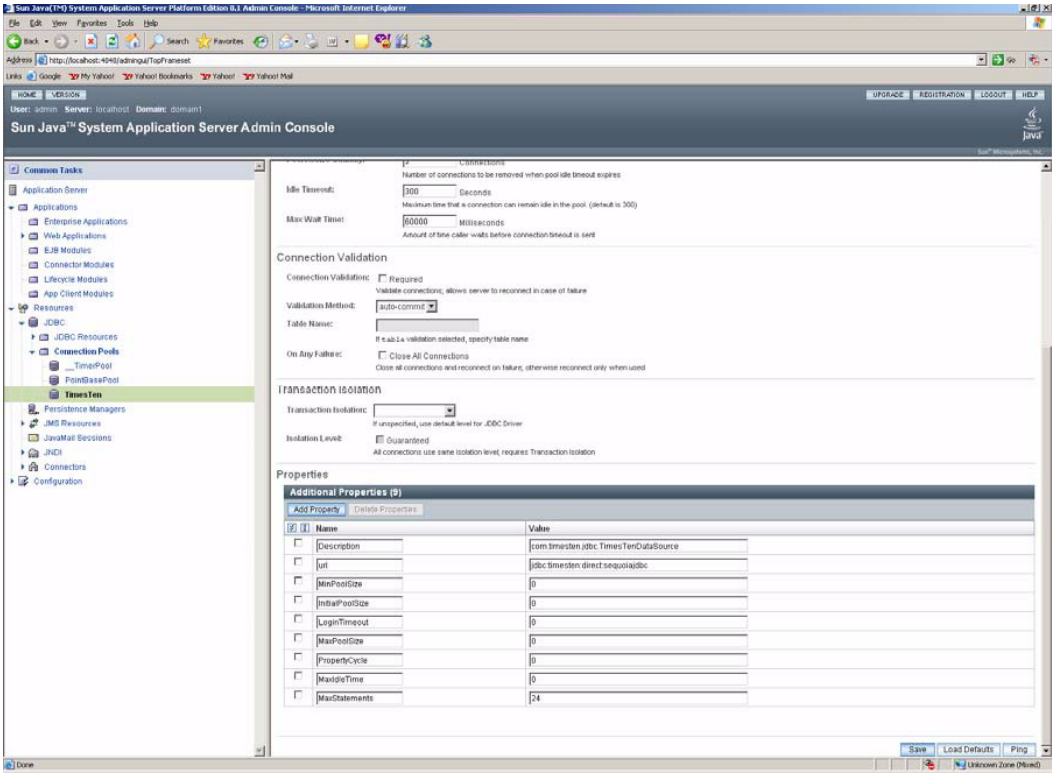
   b. Specify the property attributes, as shown in Figure 4.4 on page 31:

   – Name: `url`

   – Values: `jdbc:timesten:direct:ttjdbc`.

   You can populate other fields such “Transaction Isolation” based on your application’s needs.

Figure 4.4 Additional Properties

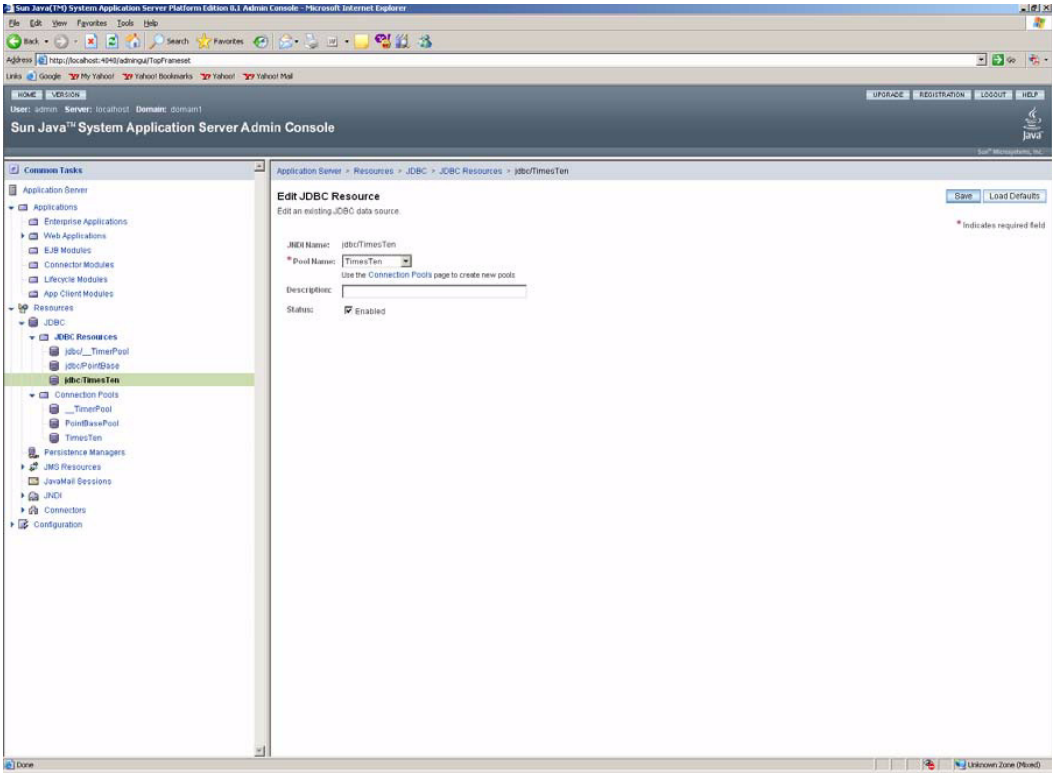


9. To verify your configuration, click **Ping**. If the configuration is working, the Admin Console will display “Ping Succeeded.”

Adding a JDBC Resource

1. Log into the Sun Java System Application Server Admin Console.
2. Navigate to **Resources -> JDBC -> JDBC Resources**.
3. Click **New**.
4. Specify the attributes for the new resource, as shown in Figure 4.5 on page 32:
   - JNDI Name: `jdbc/TimesTen`
   - Pool Name: `TimesTen`
5. Click **OK**. This adds a JNDI resource called `jdbc/TimesTen` and associates it with the previously created TimesTen pool. Now you are ready to use the ConnectionPoolDataSource called `jdbc/TimesTen`.

Figure 4.5 JDBC Resource



5

Configuring WebSphere and TimesTen

This document provides descriptions and examples of configuring the TimesTen JDBC driver for use with IBM's WebSphere Application Server 6 The document assumes that the reader has a basic familiarity with the use of both WebSphere and TimesTen Data Server.

This chapter includes the following topics:

Supported JDBC Driver Configurations

WebSphere 6.0 can be used with four distinct TimesTen JDBC driver configurations. Use the following table as a guide when configuring WebSphere JDBC providers and data sources for TimesTen.

Table 5.1 Supported JDBC driver configurations

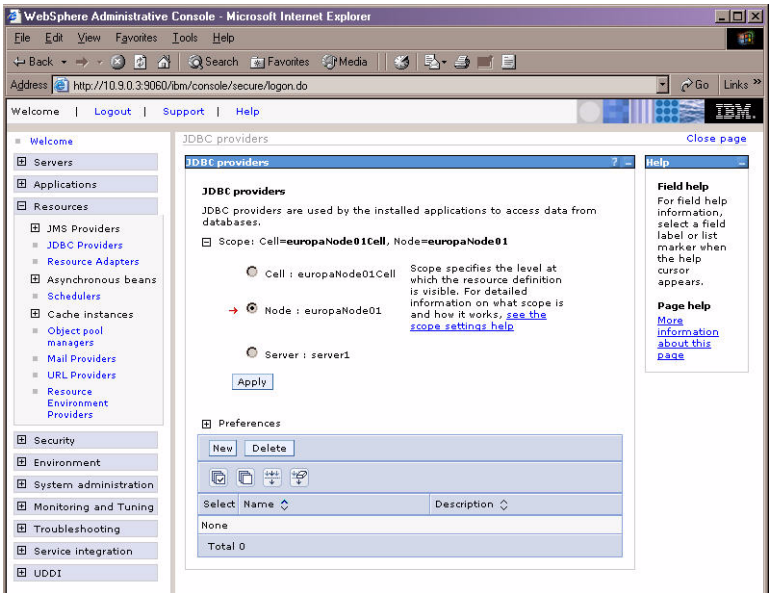
TimesTen Driver Type	Driver Class Name	Example URL
Direct	`com.timesten.jdbc.ObservableConnectionDS`	`jdbc:timesten:MYDSN`
Client-server	`com.timesten.jdbc.ObservableConnectionDS`	`jdbc:timesten:client:MYDSNCS`
Direct XA	`com.timesten.jdbc.xa.TimesTenXADataSource`	`jdbc:timesten:MYDSN`
Client-server XA	`com.timesten.jdbc.xa.TimesTenXADataSource`	`jdbc:timesten:client:MYDSNCS`

Configuring the TimesTen JDBC Provider

Perform the following steps to configure TimesTen as a JDBC provider in WebSphere:

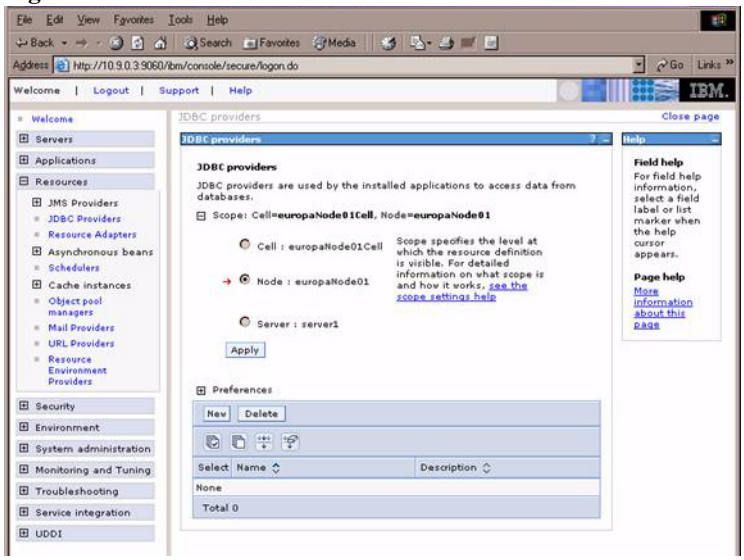
1. From the WebSphere Administration Console select **Resources** > **JDBC Providers**.
2. On the **JDBC Providers** page, select a scope for the TimesTen driver configuration.
3. Click **New**. See Figure 5.1.

Figure 5.1 JDBC Providers



4. On the **JDBC Providers > New** page, select **User-defined** for the database type.
5. Click **Next**. See Figure 5.2.

Figure 5.2 JDBC Providers > New



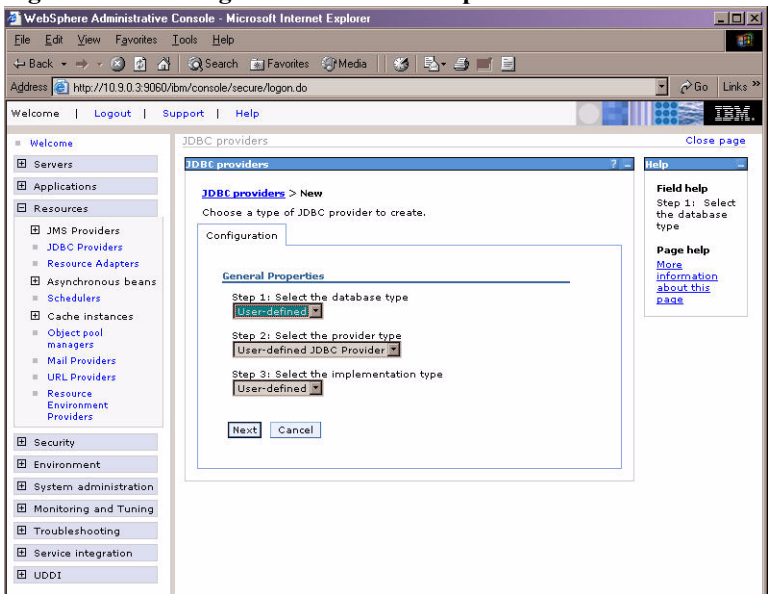
6. Type a name for the provider and an optional description.
7. In the **Class path** section, specify the full path to the TimesTen JDBC driver jar file. For example: `/home/timesten/TimesTen/tt60/lib/classes14.jar`
8. In the **Native library path** section, specify the path of the `lib` directory of the TimesTen installation. For example: `/home/timesten/TimesTen/tt60/lib`

Note: For a complete description of the environment variable requirements for a particular platform, please see the "Environment modifications" section of the *Oracle TimesTen In-Memory Database Installation Guide*.

---

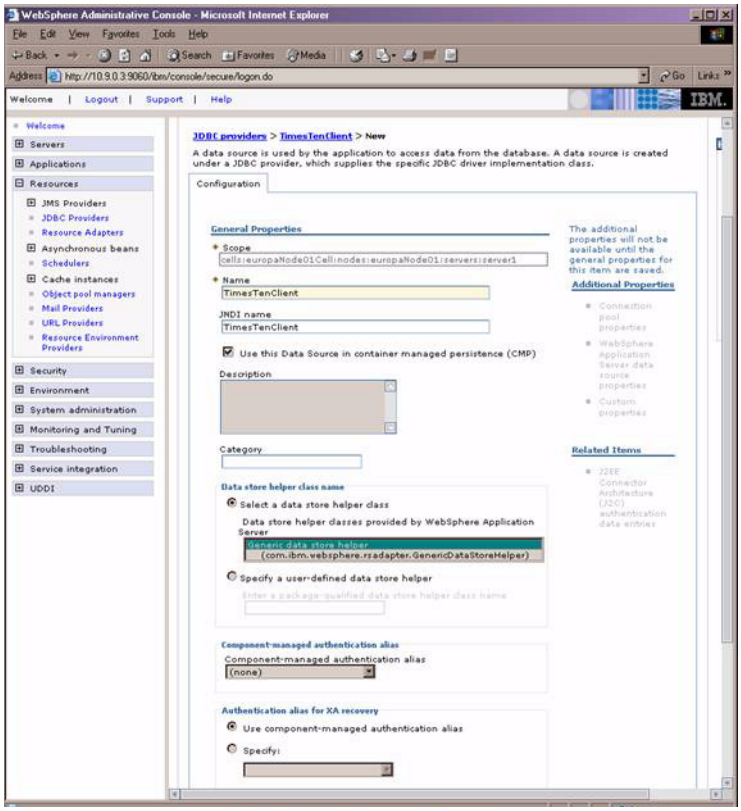
9. In the **Implementation class name** section, enter the name of the TimesTen JDBC driver class. For direct or client-server connections with local transactions use `com.timesten.jdbc.ObservableConnectionDS`. If XA transactions are required, then use `com.timesten.jdbc.xa.TimesTenXADataSource`.
10. Click **Apply**. See Figure 5.3.

Figure 5.3 Configuration General Properties



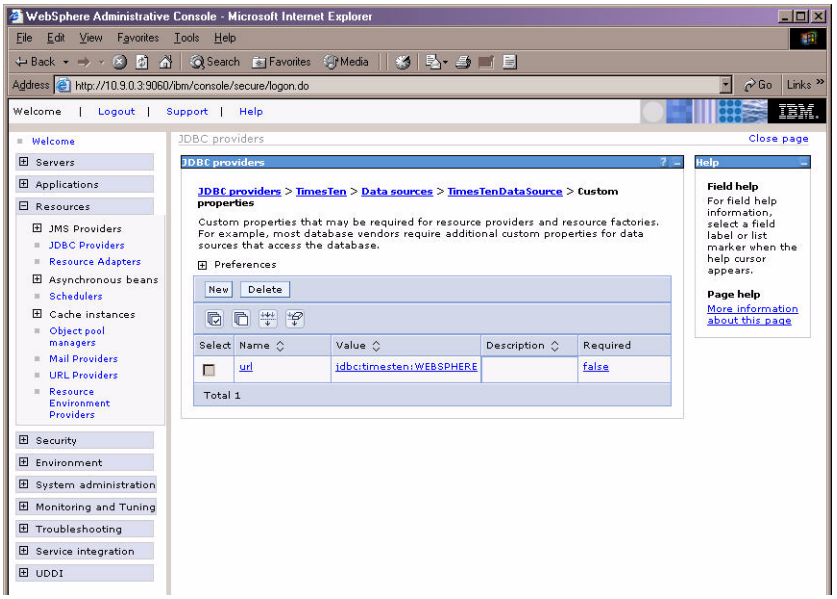
11. A **Data sources** link appears on the right hand side of the **JDBC Providers > New** page. Click **Data sources** and then click **New**.
12. Enter a name for the data source in the **Name** field.
13. Enter the JNDI name of the Data source in the **JNDI name** field.
14. In the **Data store helper class name** section, select **Generic data store helper**.
15. Click **Apply** at the bottom of the page. See Figure 5.4.

Figure 5.4 Data Sources



16. A **Custom properties** link appears on the right side of the page. Click **Custom properties**.
17. Click **New** to create a new property.
18. Every TimesTen data store in WebSphere must have a custom String property called '`url`'. The value of this property is a valid TimesTen URL that identifies the DSN to connect to. For example, direct access to a DSN called WEBSPHERE can be specified as '`jdbc:timesten:WEBSPHERE`'.
19. Click **Apply** to register the property. See Figure 5.5.

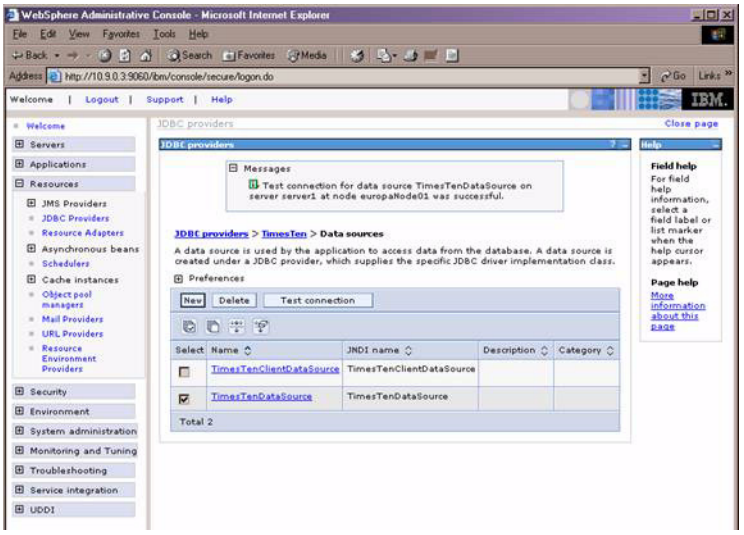
Figure 5.5 Custom Properties



20. At this point the TimesTen JDBC provider and a data source have been configured in WebSphere. The WebSphere Administration Console prompts to save these changes. Save the changes, log out, and restart the application server.

21. To test the TimesTen data source connection, log into the Administration Console and select **Resources** > **JDBCProviders**. Select the name of the TimesTen provider that was configured in the previous sections. Select **Data sources** on the right side. Select the TimesTen data source to test and then click the **Test Connection** button. If the connection fails, then a link to the server's log appears with diagnostic information. See Figure 5.6.

Figure 5.6 Test Connection



A

Sample Configuration Files for JBoss

This appendix contains the sample configuration files described in "Configuring JBoss and TimesTen" on page 15:

- **timesten-ds.xml**
- **timesten-jdbc2-service.xml**
- **timesten-jdbc-state-service.xml**

timesten-ds.xml

Example 5.1 shows a sample TimesTen datasource configuration file.

Example 5.1

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ===================================================================== -->
<!--                                                                       -->
<!--  TimesTen datasource configurations                                   -->
<!--                                                                       -->
<!-- ===================================================================== -->


<!-- TimesTen local transaction datasources -->

<datasources>

  <local-tx-datasource>
    <jndi-name>TimesTenLocalDS</jndi-name>
    <connection-url>jdbc:timesten:JBOSS</connection-url>
    <driver-class>com.timesten.jdbc.TimesTenDriver</driver-class>

    <user-name>jboss</user-name>
    <password>jboss</password>
    <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>

    <!--pooling parameters-->
    <min-pool-size>5</min-pool-size>
    <max-pool-size>100</max-pool-size>
    <blocking-timeout-millis>5000</blocking-timeout-millis>
```

```
  <idle-timeout-minutes>15</idle-timeout-minutes>

  <prepared-statement-cache-size>32</prepared-statement-cache-size>

  <!-- sql to call when connection is created -->
  <new-connection-sql>SELECT * FROM SYS.TABLES</new-connection-sql>

 <!-- sql to call on an existing pooled connection when it is obtained from pool -->
  <check-valid-connection-sql>
     SELECT * FROM SYS.TABLES
  </check-valid-connection-sql>

  <!-- corresponding type-mapping in the standardjbosscmp-jdbc.xml -->
  <metadata>
     <type-mapping>TimesTen</type-mapping>
  </metadata>
</local-tx-datasource>


<local-tx-datasource>
  <jndi-name>TimesTenLocalClientDS</jndi-name>
  <connection-url>jdbc:timesten:client:JBOSS_CS</connection-url>
  <driver-class>com.timesten.jdbc.TimesTenDriver</driver-class>

  <user-name>jboss</user-name>
  <password>jboss</password>
  <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>

  <!--pooling parameters-->
  <min-pool-size>5</min-pool-size>
  <max-pool-size>100</max-pool-size>
  <blocking-timeout-millis>5000</blocking-timeout-millis>
  <idle-timeout-minutes>15</idle-timeout-minutes>

  <prepared-statement-cache-size>32</prepared-statement-cache-size>


  <!-- sql to call when connection is created -->
  <new-connection-sql>SELECT * FROM SYS.TABLES</new-connection-sql>

 <!-- sql to call on an existing pooled connection when it is obtained from pool -->
  <check-valid-connection-sql>
     SELECT * FROM SYS.TABLES
  </check-valid-connection-sql>

  <!-- corresponding type-mapping in the standardjbosscmp-jdbc.xml -->
  <metadata>
     <type-mapping>TimesTen</type-mapping>
  </metadata>
```

```
</local-tx-datasource>


<!-- TimesTen XA transaction datasources -->

<xa-datasource>
  <jndi-name>TimesTenXADS</jndi-name>
  <xa-datasource-class>
     com.timesten.jdbc.xa.TimesTenXADataSource
  </xa-datasource-class>
  <xa-datasource-property name="Url">
     jdbc:timesten:JBOSS
  </xa-datasource-property>

  <user-name>jboss</user-name>
  <password>jboss</password>
  <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>

  <!--pooling parameters-->
  <min-pool-size>5</min-pool-size>
  <max-pool-size>100</max-pool-size>
  <blocking-timeout-millis>5000</blocking-timeout-millis>
  <idle-timeout-minutes>15</idle-timeout-minutes>

  <prepared-statement-cache-size>32</prepared-statement-cache-size>


  <!-- This is required by TimesTen XA data sources. If it is not included
       then XA transactions can fail with various transaction management
      errors including javax.transaction.xa.XAException: errorCode=XAER_PROTO
  -->
  <track-connection-by-tx/>


  <!-- sql to call when connection is created -->
  <new-connection-sql>SELECT * FROM SYS.TABLES</new-connection-sql>

 <!-- sql to call on an existing pooled connection when it is obtained from pool -->
  <check-valid-connection-sql>
     SELECT * FROM SYS.TABLES
  </check-valid-connection-sql>

  <!-- corresponding type-mapping in the standardjbosscmp-jdbc.xml -->
  <metadata>
     <type-mapping>TimesTen</type-mapping>
  </metadata>
</xa-datasource>
```

```
<xa-datasource>
  <jndi-name>TimesTenXAClientDS</jndi-name>
  <xa-datasource-class>
     com.timesten.jdbc.xa.TimesTenXADataSource
  </xa-datasource-class>
  <xa-datasource-property name="Url">
     jdbc:timesten:client:JBOSS_CS
  </xa-datasource-property>

  <user-name>jboss</user-name>
  <password>jboss</password>
  <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>

  <!--pooling parameters-->
  <min-pool-size>5</min-pool-size>
  <max-pool-size>100</max-pool-size>
  <blocking-timeout-millis>5000</blocking-timeout-millis>
  <idle-timeout-minutes>15</idle-timeout-minutes>

  <prepared-statement-cache-size>32</prepared-statement-cache-size>


  <!-- This is required by TimesTen XA data sources. If it is not included
       then XA transactions can fail with various transaction management
      errors including javax.transaction.xa.XAException: errorCode=XAER_PROTO
  -->
  <track-connection-by-tx/>

  <!-- sql to call when connection is created -->
  <new-connection-sql>SELECT * FROM SYS.TABLES</new-connection-sql>

 <!-- sql to call on an existing pooled connection when it is obtained from pool -->
  <check-valid-connection-sql>
     SELECT * FROM SYS.TABLES
  </check-valid-connection-sql>

  <!-- corresponding type-mapping in the standardjbosscmp-jdbc.xml -->
  <metadata>
     <type-mapping>TimesTen</type-mapping>
  </metadata>
</xa-datasource>

<!-- TimesTen persistent JMS data source -->

<xa-datasource>
  <jndi-name>TimesTenJMS</jndi-name>
  <xa-datasource-class>
     com.timesten.jdbc.xa.TimesTenXADataSource
```

```
    </xa-datasource-class>
    <xa-datasource-property name="Url">
       jdbc:timesten:JBOSS_JMS
    </xa-datasource-property>

    <user-name>jboss</user-name>
    <password>jboss</password>
    <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>

    <!--pooling parameters-->
    <min-pool-size>5</min-pool-size>
    <max-pool-size>100</max-pool-size>
    <blocking-timeout-millis>5000</blocking-timeout-millis>
    <idle-timeout-minutes>15</idle-timeout-minutes>

    <prepared-statement-cache-size>32</prepared-statement-cache-size>


    <!-- This is required by TimesTen XA data sources. If it is not included
         then XA transactions can fail with various transaction management
        errors including javax.transaction.xa.XAException: errorCode=XAER_PROTO
    -->
    <track-connection-by-tx/>


    <!-- sql to call when connection is created -->
    <new-connection-sql>SELECT * FROM SYS.TABLES</new-connection-sql>

   <!-- sql to call on an existing pooled connection when it is obtained from pool -->
    <check-valid-connection-sql>
       SELECT * FROM SYS.TABLES
    </check-valid-connection-sql>

    <!-- corresponding type-mapping in the standardjbosscmp-jdbc.xml -->
    <metadata>
       <type-mapping>TimesTen</type-mapping>
    </metadata>
  </xa-datasource>
</datasources>
```

timesten-jdbc2-service.xml

Example 5.2 shows a sample JDBC2 Service configuration for configuring TimesTen as the RDBMS for JBoss JMS.

Example 5.2

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<server>

  <!-- ===================================================================== -->
  <!-- JMS persistence and caching using TimesTen                            -->
  <!-- IMPORTANT: Remove hsqldb-jdbc2-service.xml                            -->
  <!-- ===================================================================== -->

  <!--
     | The destination manager is the core service within JBossMQ
  -->
  <mbean code="org.jboss.mq.server.jmx.DestinationManager"
      name="jboss.mq:service=DestinationManager">
    <depends optional-attribute-name="MessageCache">
      jboss.mq:service=MessageCache
    </depends>
    <depends optional-attribute-name="PersistenceManager">
      jboss.mq:service=PersistenceManager
    </depends>
    <depends optional-attribute-name="StateManager">
      jboss.mq:service=StateManager
    </depends>
  </mbean>

  <!--
     | The MessageCache decides where to put JBossMQ message that
     | are sitting around waiting to be consumed by a client.
     |
     | The memory marks are in Megabytes.  Once the JVM memory usage hits
     | the high memory mark, the old messages in the cache will start getting
     | stored in the DataDirectory.  As memory usage gets closer to the
     | Max memory mark, the amount of message kept in the memory cache
     | approaches 0.
   -->
  <mbean code="org.jboss.mq.server.MessageCache"
       name="jboss.mq:service=MessageCache">
    <attribute name="HighMemoryMark">50</attribute>
    <attribute name="MaxMemoryMark">60</attribute>
    <attribute name="CacheStore">jboss.mq:service=PersistenceManager</attribute>
  </mbean>


  <!-- The PersistenceManager is used to store messages to disk. -->
  <!--
     | The jdbc2 PersistenceManager is the new improved JDBC implementation.
     | This implementation allows you to control how messages are stored in
     | the database.
     |
     | This jdbc2 PM configuration has been tested against TimesTen
   -->
```

```
  <mbean code="org.jboss.mq.pm.jdbc2.PersistenceManager"
       name="jboss.mq:service=PersistenceManager">
    <depends optional-attribute-name="ConnectionManager">
      jboss.jca:service=DataSourceBinding,name=TimesTenJMS
    </depends>
    <attribute name="SqlProperties">
      BLOB_TYPE = BINARYSTREAM_BLOB
      INSERT_TX = INSERT INTO JMS_TRANSACTIONS (TXID) values(?)
      INSERT_MESSAGE = INSERT INTO JMS_MESSAGES (
          MESSAGEID, DESTINATION, MESSAGEBLOB, TXID, TXOP)
          VALUES(?,?,?,?,?)
      SELECT_ALL_UNCOMMITED_TXS = SELECT TXID FROM JMS_TRANSACTIONS
      SELECT_MAX_TX = SELECT MAX(TXID) FROM JMS_MESSAGES
      SELECT_MESSAGES_IN_DEST = SELECT MESSAGEID, MESSAGEBLOB FROM JMS_MESSAGES
          WHERE DESTINATION=?
      SELECT_MESSAGE = SELECT MESSAGEID, MESSAGEBLOB FROM JMS_MESSAGES WHERE
          MESSAGEID=? AND DESTINATION=?
      MARK_MESSAGE = UPDATE JMS_MESSAGES SET TXID=?, TXOP=? WHERE MESSAGEID=?
          AND DESTINATION=?
      UPDATE_MESSAGE = UPDATE JMS_MESSAGES SET MESSAGEBLOB=? WHERE MESSAGEID=?
          AND DESTINATION=?
      UPDATE_MARKED_MESSAGES = UPDATE JMS_MESSAGES SET TXID=?, TXOP=? WHERE
          TXOP=?
      UPDATE_MARKED_MESSAGES_WITH_TX = UPDATE JMS_MESSAGES SET TXID=?, TXOP=?
          WHERE TXOP=? AND TXID=?
      DELETE_MARKED_MESSAGES_WITH_TX = DELETE FROM JMS_MESSAGES MESS WHERE TXOP=?
          AND EXISTS (SELECT TXID FROM JMS_TRANSACTIONS TX WHERE TX.TXID = MESS.TXID)
      DELETE_TX = DELETE FROM JMS_TRANSACTIONS WHERE TXID = ?
      DELETE_MARKED_MESSAGES = DELETE FROM JMS_MESSAGES WHERE TXID=? AND TXOP=?
      DELETE_TEMPORARY_MESSAGES = DELETE FROM JMS_MESSAGES WHERE TXOP='T'
      DELETE_MESSAGE = DELETE FROM JMS_MESSAGES WHERE MESSAGEID=?
          AND DESTINATION=?
      CREATE_MESSAGE_TABLE = CREATE TABLE JMS_MESSAGES (
          MESSAGEID INTEGER NOT NULL, \
          DESTINATION VARCHAR(255) NOT NULL, TXID INTEGER, TXOP CHAR(1), \
          MESSAGEBLOB VARBINARY(4194304), PRIMARY KEY (MESSAGEID, DESTINATION) )
      CREATE_IDX_MESSAGE_TXOP_TXID = CREATE INDEX JMS_MESSAGES_TXOP_TXID ON
          JMS_MESSAGES (TXOP, TXID)
      CREATE_IDX_MESSAGE_DESTINATION = CREATE INDEX JMS_MESSAGES_DESTINATION ON
          JMS_MESSAGES (DESTINATION)
      CREATE_TX_TABLE = CREATE TABLE JMS_TRANSACTIONS ( TXID INTEGER, PRIMARY
          KEY (TXID) )
      CREATE_TABLES_ON_STARTUP = TRUE
    </attribute>
  </mbean>

</server>
```

timesten-jdbc-state-service.xml

Example 5.3 shows a sample JDBC State Service configuration for configuring TimesTen as the RDBMS for JBoss JMS.

Example 5.3

```
<?xml version="1.0" encoding="UTF-8"?>


<server>

  <!-- ===================================================================== -->
  <!-- JBossMQ State Management using TimesTen                               -->
  <!-- See docs/examples/jms for other configurations                        -->
  <!-- ===================================================================== -->

  <!-- A Statemanager that stores state in the database -->
  <mbean code="org.jboss.mq.sm.jdbc.JDBCStateManager"
           name="jboss.mq:service=StateManager">
    <depends optional-attribute-name="ConnectionManager">
       jboss.jca:service=DataSourceBinding,name=TimesTenJMS
    </depends>
    <attribute name="SqlProperties">
      CREATE_TABLES_ON_STARTUP = TRUE
      CREATE_USER_TABLE = CREATE TABLE JMS_USERS (USERID VARCHAR(32) NOT NULL,
          PASSWD VARCHAR(32) NOT NULL, \
          CLIENTID VARCHAR(128), PRIMARY KEY(USERID))
      CREATE_ROLE_TABLE = CREATE TABLE JMS_ROLES (ROLEID VARCHAR(32) NOT NULL,
          USERID VARCHAR(32) NOT NULL, \
          PRIMARY KEY(USERID, ROLEID))
      CREATE_SUBSCRIPTION_TABLE = CREATE TABLE JMS_SUBSCRIPTIONS (CLIENTID
          VARCHAR(128) NOT NULL, \
          SUBNAME VARCHAR(128) NOT NULL, TOPIC VARCHAR(255) NOT NULL, \
          SELECTOR VARCHAR(255), PRIMARY KEY(CLIENTID, SUBNAME))
      GET_SUBSCRIPTION = SELECT TOPIC, SELECTOR FROM JMS_SUBSCRIPTIONS WHERE
          CLIENTID=? AND SUBNAME=?
      LOCK_SUBSCRIPTION = SELECT TOPIC, SELECTOR FROM JMS_SUBSCRIPTIONS WHERE
          CLIENTID=? AND SUBNAME=?
      GET_SUBSCRIPTIONS_FOR_TOPIC = SELECT CLIENTID, SUBNAME, SELECTOR FROM
          JMS_SUBSCRIPTIONS WHERE TOPIC=?
      INSERT_SUBSCRIPTION = INSERT INTO JMS_SUBSCRIPTIONS (CLIENTID, SUBNAME,
          TOPIC, SELECTOR) VALUES(?,?,?,?)
      UPDATE_SUBSCRIPTION = UPDATE JMS_SUBSCRIPTIONS SET TOPIC=?, SELECTOR=?
          WHERE CLIENTID=? AND SUBNAME=?
      REMOVE_SUBSCRIPTION = DELETE FROM JMS_SUBSCRIPTIONS WHERE CLIENTID=?
          AND SUBNAME=?
      GET_USER_BY_CLIENTID = SELECT USERID, PASSWD, CLIENTID FROM JMS_USERS
          WHERE CLIENTID=?
      GET_USER = SELECT PASSWD, CLIENTID FROM JMS_USERS WHERE USERID=?
```

```
      POPULATE.TABLES.01 = INSERT INTO JMS_USERS (USERID, PASSWD) VALUES
          ('guest', 'guest')
      POPULATE.TABLES.02 = INSERT INTO JMS_USERS (USERID, PASSWD) VALUES
          ('j2ee', 'j2ee')
      POPULATE.TABLES.03 = INSERT INTO JMS_USERS (USERID, PASSWD, CLIENTID)
          VALUES ('john', 'needle', 'DurableSubscriberExample')
      POPULATE.TABLES.04 = INSERT INTO JMS_USERS (USERID, PASSWD) VALUES
          ('nobody', 'nobody')
      POPULATE.TABLES.05 = INSERT INTO JMS_USERS (USERID, PASSWD) VALUES
          ('dynsub', 'dynsub')
      POPULATE.TABLES.06 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('guest','guest')
      POPULATE.TABLES.07 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('j2ee','guest')
      POPULATE.TABLES.08 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('john','guest')
      POPULATE.TABLES.09 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('subscriber','john')
      POPULATE.TABLES.10 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('publisher','john')
      POPULATE.TABLES.11 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('publisher','dynsub')
      POPULATE.TABLES.12 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('durpublisher','john')
      POPULATE.TABLES.13 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('durpublisher','dynsub')
      POPULATE.TABLES.14 = INSERT INTO JMS_ROLES (ROLEID, USERID) VALUES
          ('noacc','nobody')
    </attribute>
  </mbean>

</server>
```

Index