TransactionsJTA for Tomcat User Guide

Table of Contents

Chapter 1. Introduction

Table of Contents

What is TransactionsJTA™ for Tomcat?

Basically, our products allow you to integrate your applications with one or more different databases in a cross-vendor and cross-platform way. In this special release, the integration is done in the form of a webserver that can run J2EE™ applications. This means that you can build a J2EE™ application that shows your different databases as one virtual system to all users. Even in the case of crashes, our software will automatically make sure that your databases remain correct and in agreement with each other.

More technically speaking, this TransactionsJTA™ release allows you to make web applications reliable by making all JDBC™ accesses transactional. This means that the total effects of a problematic web request can be cancelled, even if there is a server crash in the middle. No corrupted data nor purchases that are processed only half-way through. The benefit for you is your data quality, plus the ability to use one of the most popular J2EE™ application design patterns: the Data Access Object (DAO). The benefit for your web site visitors is the increased trust they will give you (nobody likes getting a bill for goods when the purchase failed). Many if not all of today's open-source J2EE™ platforms only offer transactions in theory but not in practice due to incomplete implementations of the JTA™ specifications. With TransactionsJTA™, you get a full JTA™ implementation including restart and crash recovery. In other words, you get a first-class transaction manager for your Tomcat webserver. At only a fraction of the cost of a 'full' J2EE™ server platform you get the same or even better reliability!

How Does It Work?

This product consists of a transaction engine that co-operates closely with the included Atomikos database connectors (JDBC-compliant). Applications that use these JDBC connectors are 'registered' with the transaction engine (this happens behind the scenes). This means that the transaction engine is aware of all JDBC-interactions and also knows about all the back-end databases involved. Whenever a request cannot complete successfully, its effects on the databases can be cancelled by the transaction core. This cancellation can be either explicit (at the request of a transactional application) or implicit (after a timeout). The scope of cancellation is called a transaction, started through the javax.transaction.UserTransaction (more on that in the rest of this guide). Applications either start such transactions explicitly (if they were developed for this) or implicitly (by addition of a transaction filter that starts and ends transactions for each request).

Atomikos TransactionsJTA for Tomcat: Unique Selling Points

Atomikos has included several patent-pending features and other strong points that are important for your J2EE projects:

For Developers

1. Developer licenses are FREE!
2. Access more than one connection in the same transaction.
3. Use the Data Access Object (DAO) pattern in its full strength.
4. EAI: develop transactional J2EE without the need for EJB.
5. Use your favorite J2EE platform.
6. You can even develop transactions in J2SE applications with the JTA library. For more information, download our TransactionsJTA standalone release.
7. Optimize performance with the intra-VM operation and advanced JTA: transactions don't have to be slower than straight-on JDBC.
8. Easily configure and check your Tomcat server for JDBC and JTA.

For Administrators

1. Maximize reliability with full recovery in place. Don't lose important data after a crash.
2. Use our graphical administration tool: the Atomikos Control Panel.
3. Automatic validation of your drivers' configuration: check if your J2EE applications are transactional, and whether you are using the right connector drivers to ensure maximum reliability. Don't wait until after a crash to find out that your drivers were not configured correctly.
4. Automatic validation of your J2EE applications' configuration: check if all required connectors are defined, and if transactions are enabled for your application.
5. Easy configuration of connectors to your back-end systems.
6. Clear overview of problematic transactions.

For Managers

1. Integrate multiple databases: cross-platform, cross-vendor, cross-database.
2. Maximize customer satisfaction by keeping their information correct at all times.
3. Minimize administration cost by automatic crash recovery that keeps your databases correct and in agreement.
4. Lower project training costs: developers have no need to know EJB.
5. Get the project done faster: the required skills are easier to find.
6. Lower hardware costs thanks to the relatively low footprint of the server pack.
7. Optimize performance with the fast, intra-VM operation.
8. Quality assurance: largely automatic validation of your EAI configuration.
9. Avoid vendor lock-in: stay J2EE standards compliant.
10. Lower license costs when compared to competing platforms.

Do I Really Need TransactionsJTA™?

After all, this would only be of benefit if there are occasional crashes, right?
Not really! Firstly, any sophisticated J2EE™ web application that is based on the Data Access Object (DAO) design pattern can only work if there is a JTA™ implementation in the server. Secondly, crashes are not that occasional at all, and tend to happen at the worst time: when the server is heavily loaded. This means that most crashes happen at the time when most visitors are on your site. In other words, if you have a crash then you can be almost certain that it affects a maximum number of visitors! If you don't have TransactionsJTA™, then you risk having to find out manually which of your 1000 visitors' payments have to be made or not... According to recent studies, the annual cost of web server crashes was as high as 128 million pounds in the UK alone.

What about Alternatives?

Open-Source

At the time of writing, there is no open-source JTA™ that can do recovery, meaning that server crashes can not be dealt with by open-source J2EE™ products. Your worst-case scenario is not covered by open-source. In addition, and paradoxically perhaps, the total cost of a typical J2EE™ project is lower with Atomikos than with the most popular open-source alternatives!

Commercial Application Servers

Most competing full commercial application servers can also offer transactional guarantees, but at a much higher cost:

1. Higher hardware cost: typically, much more memory, CPU and disk space are needed than for Atomikos.
2. More skills required: most application servers are difficult to use and program.
3. Lots of unused features: only a very small fraction of all J2EE projects uses Enterprise JavaBeans (EJBs).

This release of TransactionsJTA™ provides most of what these other products offer but at a fraction of the hardware cost (low footprint), very easy to use, and easy to program (no EJBs which prove obsolete for many projects). In other words, with TransactionsJTA™, you get a powerful and state-of-the-art transaction platform for your webserver.

In summary, our studies show that the total costs (software, infrastructure, training and services) of most J2EE™ solutions (including open-source!) surpass Atomikos' total cost, whereas the quality they offer is usually lower.

Chapter 2. Installing TransactionsJTA™ for Tomcat

Table of Contents

Read this chapter to setup and verify your installation.

Minimum System Requirements

The following are the minimum requirements for your system to be able to use this software. Most modern desktop computers should be more than sufficient.

1. A Java J2SDK 1.4.1 or higher (freely available at http://java.sun.com)
2. 100 MB of free hard disk space
3. 128 MB of RAM
4. An ethernet card
5. A web browser

Installing the Preconfigured J2EE™ Webserver Pack

To install the preconfigured combination (TransactionsJTA™ and Tomcat) you only need to unzip the .zip archive file as obtained from download or from the installation CD. Unzip to the folder of your preference, called JTA_INSTALLATION_FOLDER in the rest of this manual.

Necessary System Environment Settings

The following sections explain how to make sure that your system is properly configured. Please follow these steps, or startup may fail.

Verify Your Java Version

Open a command (shell) window and type: 'java -version'. The result should show at least 1.4.x. If not, please install a newer Java J2SDK release. Close the command window again.

Set the JAVA_HOME Environment Variable

1. On Windows, open the System control panel, and go to the tab 'Advanced'. Click on 'Environment Variables' and make sure that JAVA_HOME is there and points to your latest java J2SDK installation folder.

   #### Caution

Make sure that the spelling is JAVA_HOME (upper case and underscore). Also, make sure to point to the top-level folder of your java (don't include the bin folder!)

2. On Linux/Unix, open a shell window and type 'echo $JAVA_HOME' and verify that this points to the correct java installation directory. If not, consult you shell-specific documentation on how to set this variable.
3. On Mac OSX, open a terminal window and type 'echo $JAVA_HOME'. The value should be /usr (if not, type 'setenv JAVA_HOME /usr' or add this to your .login file).

Verify Port Usage

Before starting up, please make sure that no other program is using the port 8080. To do so, open a terminal/ command window and type 'netstat -a' (which works on virtually all systems). Assert that the 'Local Address' column does not show any listing that involves 8080.

Assert Script Execution Mode

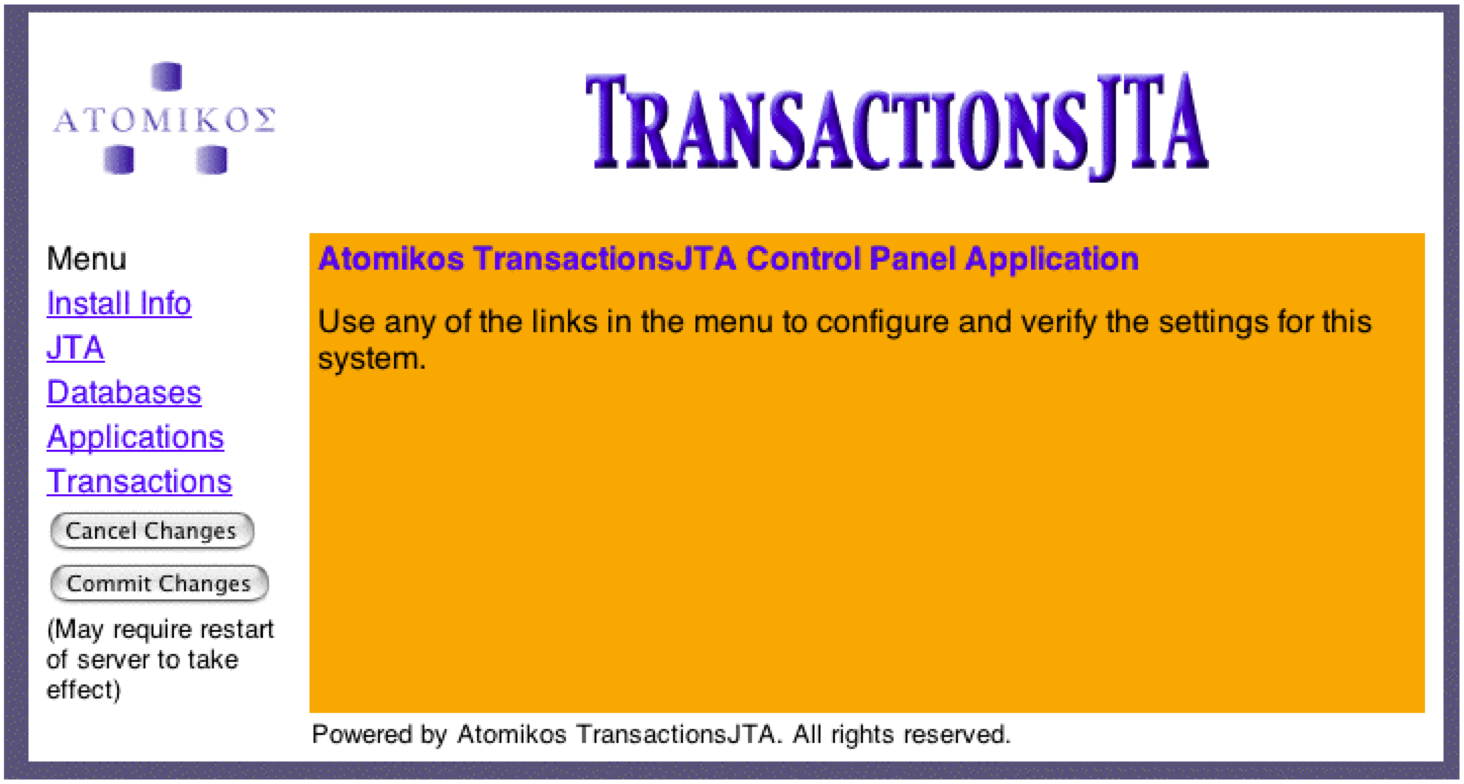
On Linux/UNIX and Mac OSX, the scripts may need to be explicitly enabled for running. To do so, open a shell/ terminal window, go into the bin folder of the installation and type: 'chmod u+x *.sh' to enable scripts for running.

Starting and Verifying Your Installation

Your system should now be ready for running the server. To verify that the installation worked, do the following:

1. Execute JTA_INSTALLATION_FOLDER/tomcat/bin/startup.bat (startup.sh on UNIX systems). Startup will fail if you are using a non-supported Java release. NOTE: you may see some errors caused by the pre-installed database connectors which try to connect to the FirstSQL database (which is not installed yet). Ignore these for now.
2. Open your web browser.
3. Enter the address http://localhost:8080/atomikos
4. Login as administrator (the default admin user is tomcat/tomcat).

You should now see the Atomikos Control Panel application:



Atomikos TransactionsJTA Control Panel Application

Use any of the links in the menu to configure and verify the settings for this system.

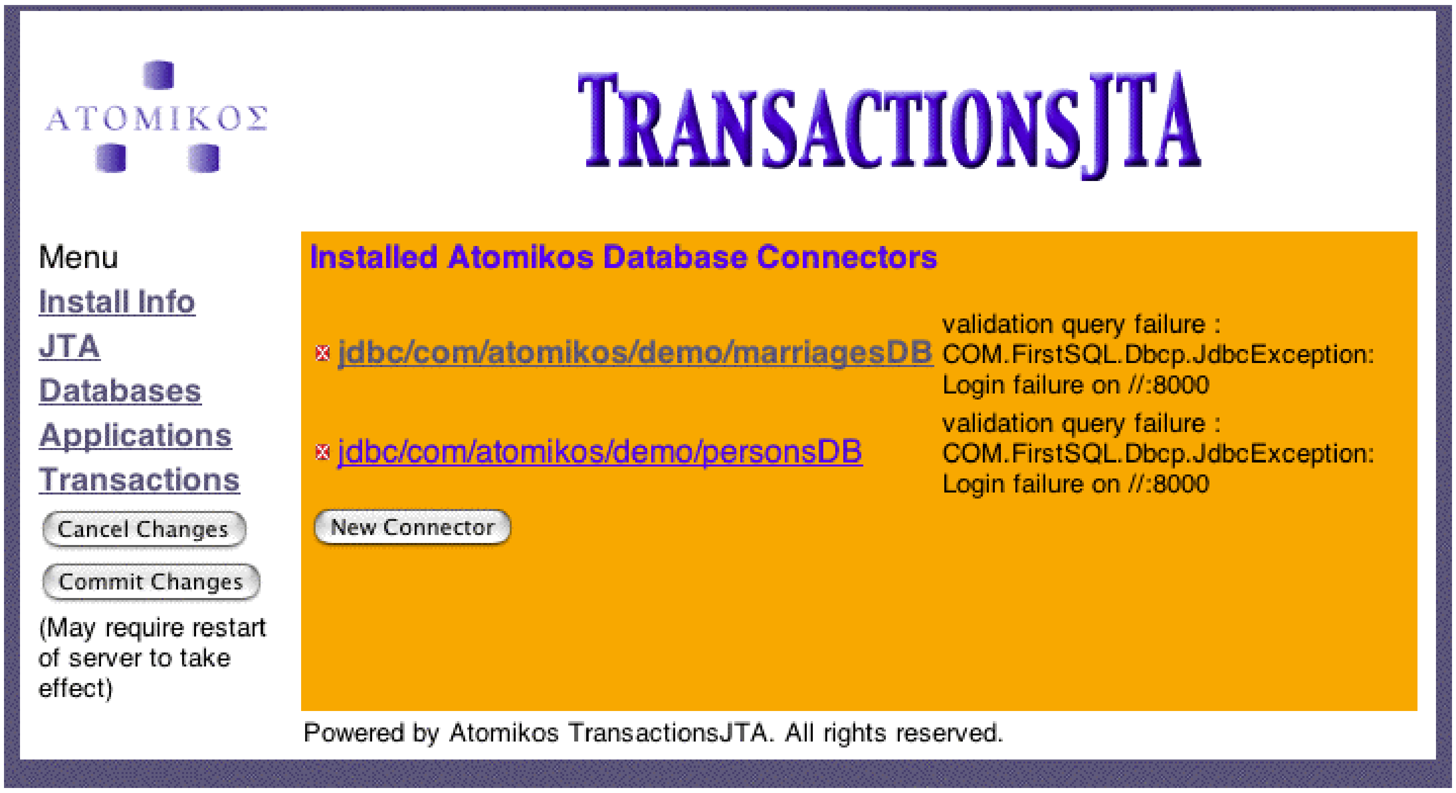
Powered by Atomikos TransactionsJTA. All rights reserved.

Caution

You should also check the 'Install Info' to make sure you have a valid license. Without a license, the product will not work.

Caution

The default installation includes two preconfigured database connectors (accessible via the 'Databases' screen). By default, these will show error messages until you install the FirstSQL software included in the release (see the picture below). These connectors are only used in the demo application (egov) and if you don't need this demo then you can delete them.



(May require restart of server to take effect)

Installed Atomikos Database Connectors

jdbc/com/atomikos/demo/marriagesDB	validation query failure : COM.FirstSQL.Dbcp.JdbcException: Login failure on //:8000
jdbc/com/atomikos/demo/personsDB	validation query failure : COM.FirstSQL.Dbcp.JdbcException: Login failure on //:8000

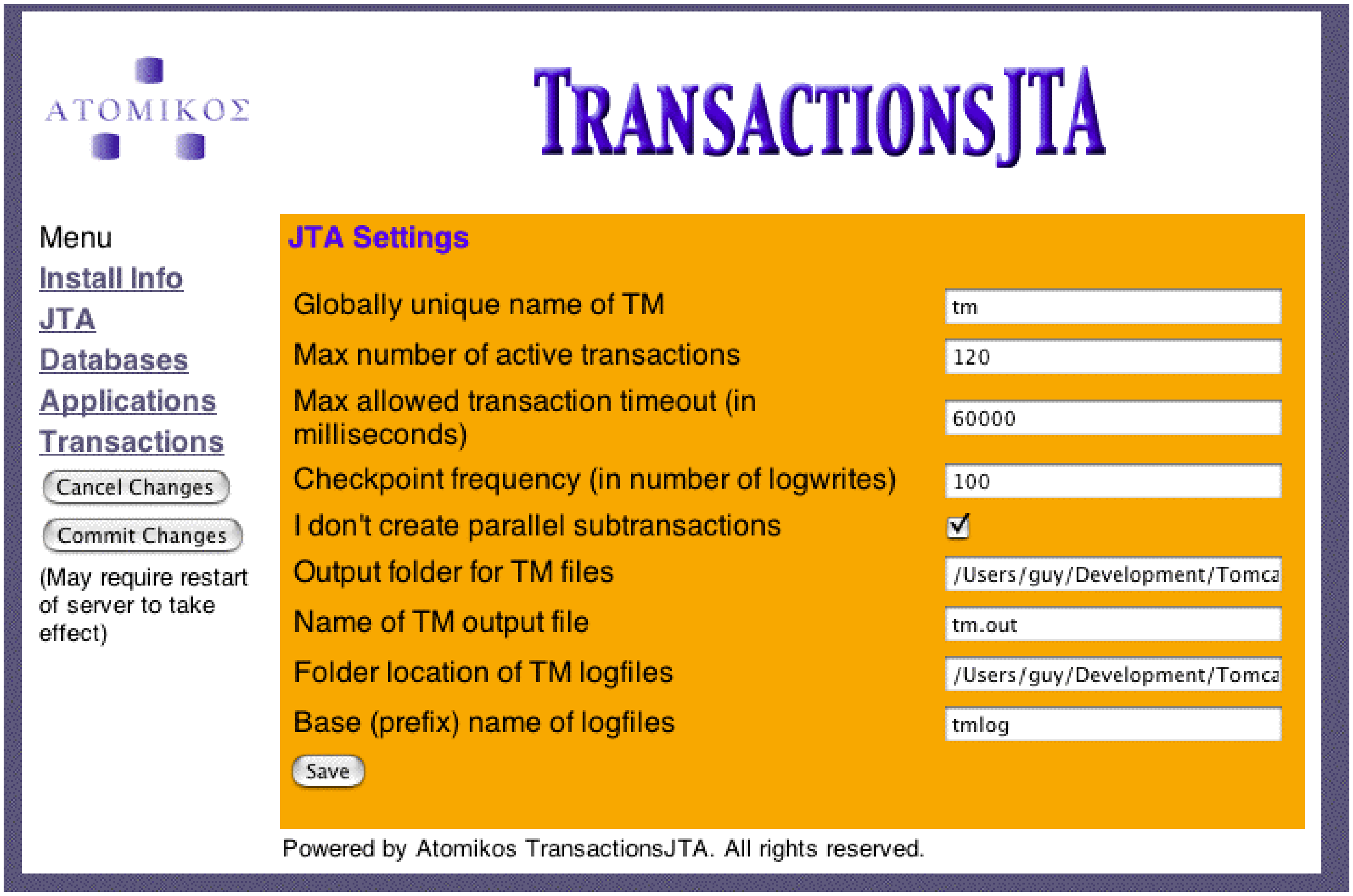
Powered by Atomikos TransactionsJTA. All rights reserved.

Verifying that JTA works

Additional verification is done through the testJTA application: http://localhost:8080/testJTA/ allows you to verify whether the transaction engine is running. Clicking any of the links will start a JTA transaction and shows a confirmation page.

Customize Your JTA™ Properties

Once it works, you should customize the JTA™ settings for your server. Doing this ensures that TransactionsJTA™ is used in the proper way and that the data in your databases remains correct at all times. Click on the menu item 'JTA' to open the right screen. An explanation of these settings follows:



JTA Settings

Globally unique name of TM	tm
Max number of active transactions	120
Max allowed transaction timeout (in milliseconds)	60000
Checkpoint frequency (in number of logwrites)	100
I don't create parallel subtransactions	✓
Output folder for TM files	/Users/guy/Development/Tomca
Name of TM output file	tm.out
Folder location of TM logfiles	/Users/guy/Development/Tomca
Base (prefix) name of logfiles	tmlog

Globally unique name of TM

A globally unique name for this server.

Caution

The value you choose is very important since it will be used to ensure that each transaction gets a unique name.

Without unique names databases can get confused about what they are doing, so it is very important that you make sure that no two different servers are using the same name for this field! We recommend that you adopt a different DNS-based name for each server you install. NOTE: if you are installing a clustered webfarm with TransactionsJTA™, then please also make sure that each server in the cluster has a different name for this field!

Max number of active transactions

Limits the number of concurrently active transactions. By choosing this value appropriately, you can control the number of users that are accessing your website. If this limit is reached, further requests will be rejected. Indirectly, this number also controls the number of concurrent database accesses allowed.

Caution

Depending on your license, this value may be limited to a lower value than you wish. Check 'Install Info' to verify if a limit is set by your license. If so, you can contact sales@atomikos.com to obtain a license upgrade if required.

Max allowed transaction timeout (in milliseconds)

Each transaction has a timeout after which it is automatically cancelled (rolled back). Web applications can specify a custom value for this timeout, but this setting limits that time interval so that transactions will never take longer than the value specified. Lower is usually better, but keep in mind that some database operations may take a long time to complete: setting this value too low will mean that longer queries or updates can never succeed since the system will cancel them each time.

Checkpoint frequency (in number of logwrites)

TransactionsJTA™ keeps logfiles that reflect the outcome of ongoing transactions. Logfiles are essential for ensuring restart or crash recoverability, and they keep on growing until the system makes a checkpoint. By choosing a low value here, you will use less disk space for the logs at the expense of more overhead (each time a checkpoint is made, time is consumed by the server).

I don't create parallel subtransactions

Check this if you want the default J2EE™ behaviour (no multi-threaded transactions). This should be sufficient for most or all J2EE™ applications, because J2EE™ does not allow multi-threading in general. However, if you want to create parallel subtransactions then you should uncheck this box. Keep in mind, however, that creating parallel subtransactions requires non-J2EE™ coding of your applications.

Output folder for TM files

This value specifies the location for some of the bookkeeping files that the server will create.

Name of TM output file

The name of the file where the text messages generated by the JTA™ are put.

Folder location of TM logfiles

This value specifies the location where the logfiles are kept. Logfiles are essential for ensuring restart and crash recoverability of your transactions.

Base (prefix) name of logfiles

The base name of the logfiles. The actual name will be based on this value as a prefix, and suffixed with a value determined by the checkpointing algorithm.

Checking License Information

You can check your license information by clicking on the menu item called 'Install Info'. For evaluation licenses, the expiry date is shown.

Stopping the Server

You can stop the server by executing shutdown.bat (shutdown.sh on UNIX systems) located in the JTA_INSTALLATION_FOLDER/tomcat/bin folder

Installing Additional Software

This release includes an evaluation copy of FirstSQL™, a Java RDBMS. Please follow the installation instructions as outlined in the FirstSQL™ folder located in the 'extra' folder of TransactionsJTA™. Make sure to follow these steps:

1. Start the installer as outlined in the installation instructions.
2. Install the Enterprise Server edition of FirstSQL (needed for transactions in JDBC).
3. Make sure that you don't change the default port (8000) for the installed server.
4. To start the server, execute the script 'StartupJ2EE' (in the server folder of your FirstSQL installation).

Caution

In general, it is recommended that you start the database server before you start the webserver. This way, recovery is allowed to proceed normally.

Adding TransactionsJTA™ to an Existing Tomcat Server

Instead of using the preconfigured combination, you can also add TransactionsJTA™ to your existing Tomcat server (Tomcat5.0.18 or higher). This section outlines the steps needed to do so.

Copy the Required JAR Files

You will need to copy the following files into the common/lib folder of your Tomcat installation:

1. transactionsJTA_1_30.jar
2. jta-spec1_0_1.jar
3. jdbc2_0-stdext.jar
4. license.jar

These files are in the lib folder of your JTA_INSTALLATION_FOLDER. In addition, you will need to copy all JDBC™-specific vendor jars to access your databases.

Install the Atomikos Console Web-Application

Deploy the file JTA_INSTALLATION_FOLDER/lib/atomikos.war in your Tomcat installation. You need this web application, or the JTA™ will not be available.

Install the JTA Test Application

Deploy the file JTA_INSTALLATION_FOLDER/lib/testJTA.war in your Tomcat. This step is not strictly necessary, but it allows you to test if the JTA works by accessing http://.../testJTA on your webserver.

Create an Administrator User

Edit the file tomcat-users.xml and add an administrator user. For example, the preconfigured server pack comes with the following entries that enable the administrator role for user tomcat/tomcat:

```
<role rolename="admin"/>
<user username="tomcat" password="tomcat" roles="admin,manager,tomcat"/>
```

Enable the Tomcat Server for JTA

1. Browse to the /atomikos application, login with your administrator account and click on 'Intall Info'. By doing this, your Tomcat will be enabled for TransactionsJTA.
2. Make sure to press 'Commit Changes'.

Caution

If you omit this part then the JTA will not be enabled in your Tomcat.

Verify your Installation

Restart your webserver (required) and access http://.../testJTA. Clicking the test links on the page should work.

Chapter 3. Installing and Configuring J2EE™ Web Applications

Table of Contents

Abstract

This chapter deals with the installation and configuration of J2EE™ web applications and making them use TransactionsJTA™ for their database access. By doing so, every access to your database can be made transactional, and can be cancelled in the event of a problem.

Prerequisites

This release supports J2EE™ web applications that conform to the servlet2.3 and JSP1.2 specifications or higher. Any such J2EE™-compliant web application that can be installed on Tomcat5.0 can also be installed on TransactionsJTA™ for Tomcat. In this release of TransactionsJTA™, the following limitations apply:

1. JNDI references to javax.transaction.UserTransaction should be looked up through the JNDI name java:comp/env/UserTransaction (note the /env/ subcontext, which is not the usual JNDI name). For maximum portability, it is recommended that developers declare the JNDI name of the UserTransaction as an application parameter in the deployment descriptor, allowing the precise value to be changed per application server.
2. Currently, JTA™ transactions are only extended to JDBC™ DataSource access and have to go through one or more configured Atomikos Database Connectors.

Installing a J2EE™ Web Application

Installing a new J2EE™ web application should be done according to the standard Tomcat procedures. In your browser, open http://localhost:8080/manager/html to use the Tomcat manager tool for the installation. Note: the default username and password are tomcat/tomcat. The instructions on the screen are straightforward.

Configuring JNDI

After installing, it is absolutely essential to configure the JNDI environment of the new web application. For each JDBC™ datasource that it needs, a link has to be defined to one of the installed database connectors. If required, new database connectors have to be installed to establish this task.

Caution

Do NOT use the Tomcat database connectors, since they are unaware of TransactionsJTA™. Use the Atomikos Control Panel (http://localhost:8080/atomikos) to configure Atomikos database connectors instead.

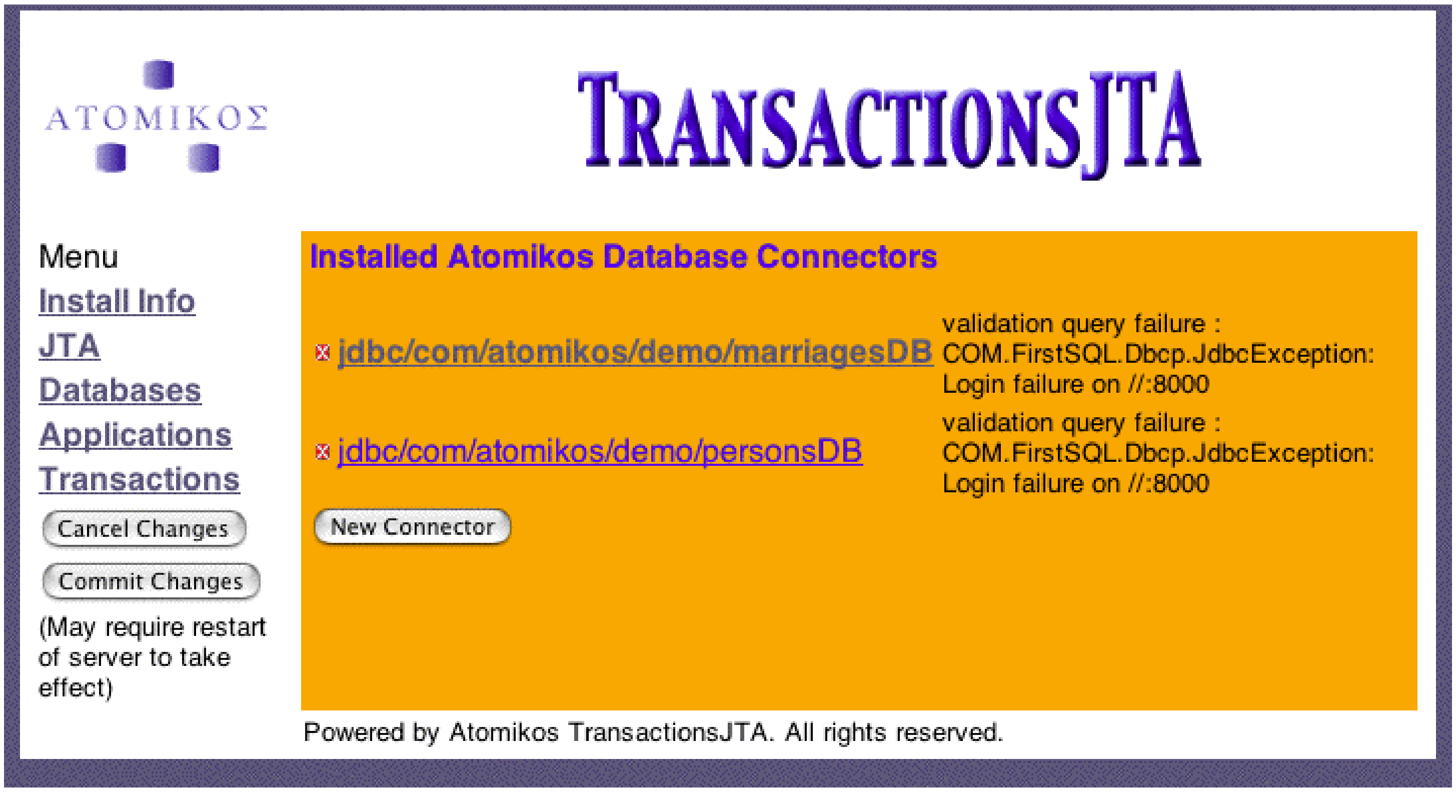
Atomikos' database connectors are transaction-aware, and can only be accessed within a transaction (comparable to the 'Required' transaction attribute of Enterprise JavaBeans).

Viewing Existing Database Connectors

If an application needs a database, then at least one database connector must exist that connects to that desired database. Database connectors are global (can be shared among different web applications) and have to be configured separately from each web application. For TransactionsJTA™, configuring these connectors should be done through the item 'Databases' from the Menu (just click on it). A screen is opened that shows all installed database connectors, if any. Clicking on a connector will open a configuration screen where you can edit the settings.

Caution

The default installation comes with two pre-installed connectors for the demo application. These connectors will NOT work until you install FirstSQL (also included in the release): the databases screen will show error messages failing to login on port 8000. If you don't want to use the pre-installed connectors then you can just delete them.



(May require restart of server to take effect)

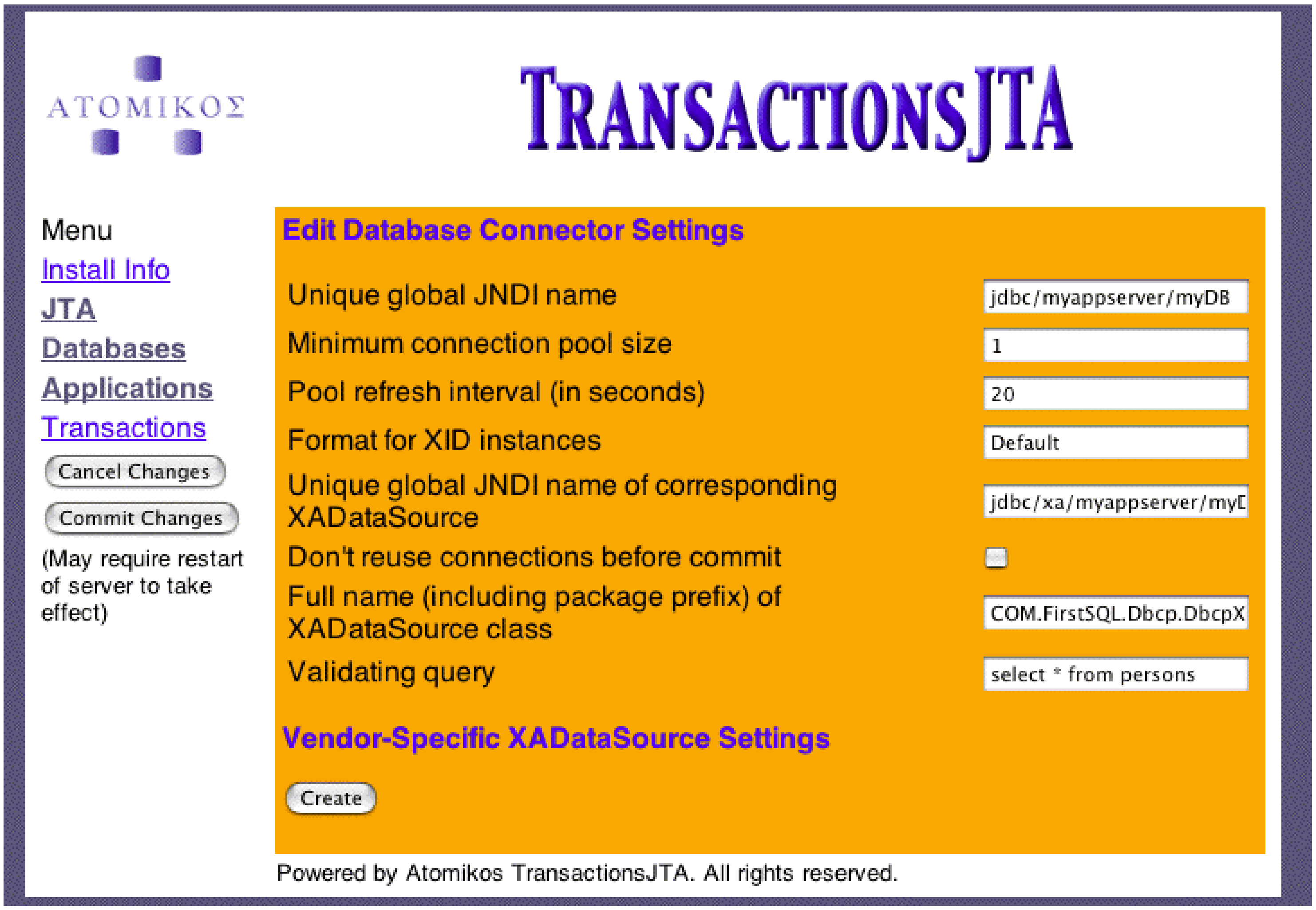
Installed Atomikos Database Connectors

jdbc/com/atomikos/demo/marriagesDB	validation query failure : COM.FirstSQL.Dbcp.JdbcException: Login failure on //:8000
jdbc/com/atomikos/demo/personsDB	validation query failure : COM.FirstSQL.Dbcp.JdbcException: Login failure on //:8000

Powered by Atomikos TransactionsJTA. All rights reserved.

Setup Database Connectors

In addition, the 'Databases' screen has a button that allows you to add new database connectors. If you click it, you will see a similar configuration screen. The configuration settings are discussed below.



Edit Database Connector Settings

Unique global JNDI name	jdbc/myappserver/myDB
Minimum connection pool size	1
Pool refresh interval (in seconds)	20
Format for XID instances	Default
Unique global JNDI name of corresponding XADataSource	jdbc/xa/myappserver/myD
Don't reuse connections before commit	☐
Full name (including package prefix) of XADataSource class	COM.FirstSQL.Dbcp.DbcpX
Validating query	select * from persons

Vendor-Specific XADataSource Settings

Unique global JNDI name

A globally unique JNDI name for the database connector.

Caution

Like the TM name, this name has to be unique across different server instances (otherwise recoverability is jeopardized).

To achieve uniqueness, we recommend that you choose a name according to the format jdbc/ unique_server_name/unique_db_name.

Minimum connection pool size

The connector will maintain a pool of connections for reuse, and this number indicates the minimum size of that pool. Note that there is no maximum pool size: if more connections are needed then more will be opened to suit that demand. If you want to limit the number of connections, you can do so through the JTA™ setting that limits the amount of active transactions.

Pool refresh interval (in seconds)

This number indicates the number of seconds between internal connection pool housekeeping tasks done by the database connector.

Format for XID instances

The database-specific format for transaction identifiers (supplied to it by the JTA™). Some databases require a custom format, but for most databases the value 'Default' will work. NOTE: for Oracle, change this value to 'Oracle'.

Unique global JNDI name of corresponding XADataSource

Every Atomikos database connector will use an underlying (vendor-specific) XADataSource driver class to connect to the database. This underlying driver is also installed in this process, and this value specifies the unique JNDI name it gets. NOTE: values should start with jdbc/xa! Hence, we recommend that you choose a name according to the format jdbc/xa/unique_server_name/db_name.

Don't reuse connections before commit

Some databases don't implement the JTA™/XA specifications correctly, and can't cope with early reuse of connections (which is the standard J2EE™ server behaviour). If you are seeing XA-specific exceptions on higher loads, you can try checking this value. Accidentally enabling this option can never be wrong, it will only decrease performance of the database connector since connection reuse will be less optimal.

Full name (including package prefix) of XADataSource class

This should be the full package-prefixed class-name of your JDBC™ vendor's XADataSource class. This class is needed by Atomikos in order to connect to the underlying database.

Caution

Please make sure that the necessary JAR files are copied into the tomcat/common/lib subfolder of your JTA_INSTALLATION_FOLDER. Tomcat requires that the driver's filename extension be .jar (not .zip). Restart the server after you copy new JAR files into common/lib.

Validating query

This field allows you to specify a SQL query that will be executed to verify your connector's configuration. Make sure to specify a query that should work under normal circumstances. TransactionsJTA™ will use this to show a warning if the query fails, allowing you to detect database connectivity problems at a glance (any such warnings will be shown on the 'Databases' screen).

Vendor-Specific XADataSource Settings

The rest of the configuration is for vendor-specific XADataSource settings, which will vary for each JDBC™ vendor and database vendor. For details, check your driver's product documentation.

Verifying Database Connectors

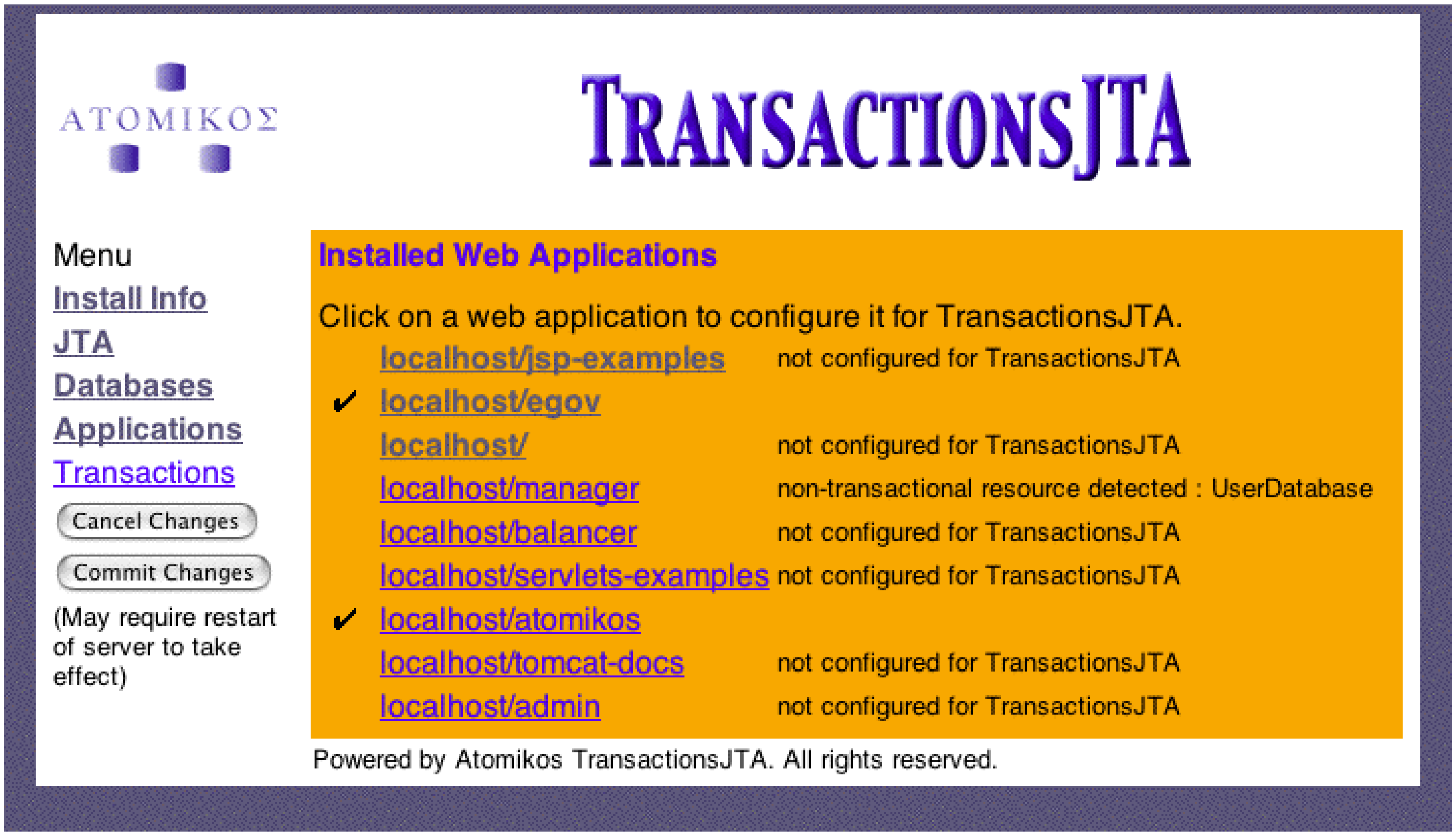
Click on 'Databases' in the menu. The screen that opens should show the status of all configured database connectors. If the system detected problems or warnings, you should see them here.

Enable the Application for JTA™

When a J2EE™ application has been installed, it needs to be enabled for TransactionsJTA™ (this is purely internal to the application server and does not modify the application in any way). In the Atomikos Control Panel, choose 'Applications' from the menu. All installed applications are shown, and by clicking on your application it can be enabled automatically for TransactionsJTA™.

Caution

Applications for which this step is omitted will NOT use TransactionsJTA™ for their transactions.



(May require restart of server to take effect)

Installed Web Applications

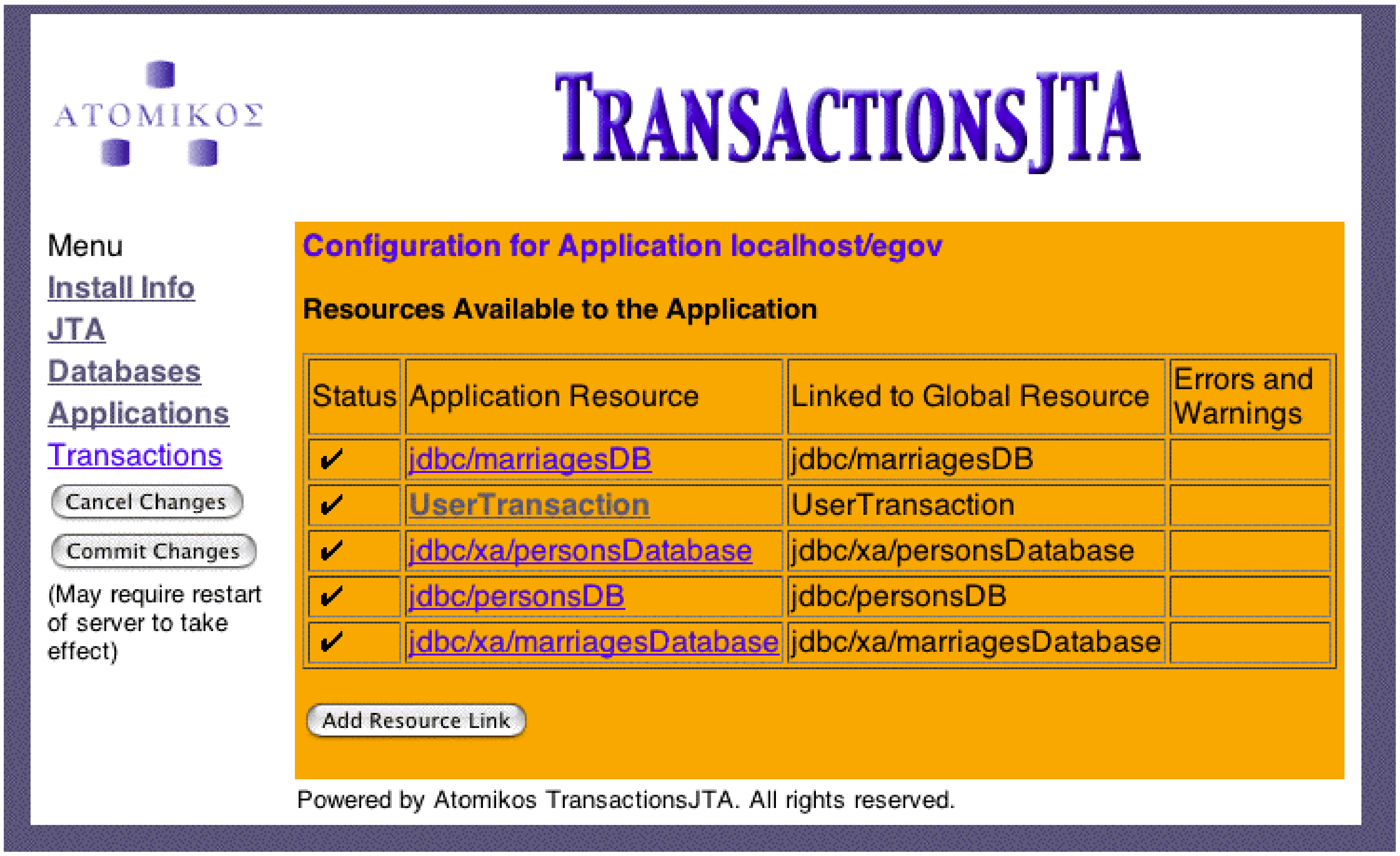
Click on a web application to configure it for TransactionsJTA.

	localhost/jsp-examples	not configured for TransactionsJTA
✔	localhost/egov	
	localhost/	not configured for TransactionsJTA
	localhost/manager	non-transactional resource detected : UserDatabase
	localhost/balancer	not configured for TransactionsJTA
	localhost/servlets-examples	not configured for TransactionsJTA
✔	localhost/atomikos	
	localhost/tomcat-docs	not configured for TransactionsJTA
	localhost/admin	not configured for TransactionsJTA

Powered by Atomikos TransactionsJTA. All rights reserved.

Create Required Links

The last remaining task is to create application-level links to any Atomikos database connectors (in Setup Database Connectors was explained that Atomikos database connectors can be shared by many web applications, but every web application needs to have a link to the installed connector first). Choose 'Applications' from the menu, and click on the application you want to create links for (if the application is not yet enabled for TransactionsJTA™, the system will propose to do so now). You will see a screen that shows all currently defined TransactionsJTA™-links for the application:



(May require restart of server to take effect)

Configuration for Application localhost/egov

Resources Available to the Application

Status	Application Resource	Linked to Global Resource	Errors and Warnings
✔	jdbc/marriagesDB	jdbc/marriagesDB	
✔	UserTransaction	UserTransaction	
✔	jdbc/xa/personsDatabase	jdbc/xa/personsDatabase	
✔	jdbc/personsDB	jdbc/personsDB	
✔	jdbc/xa/marriagesDatabase	jdbc/xa/marriagesDatabase	

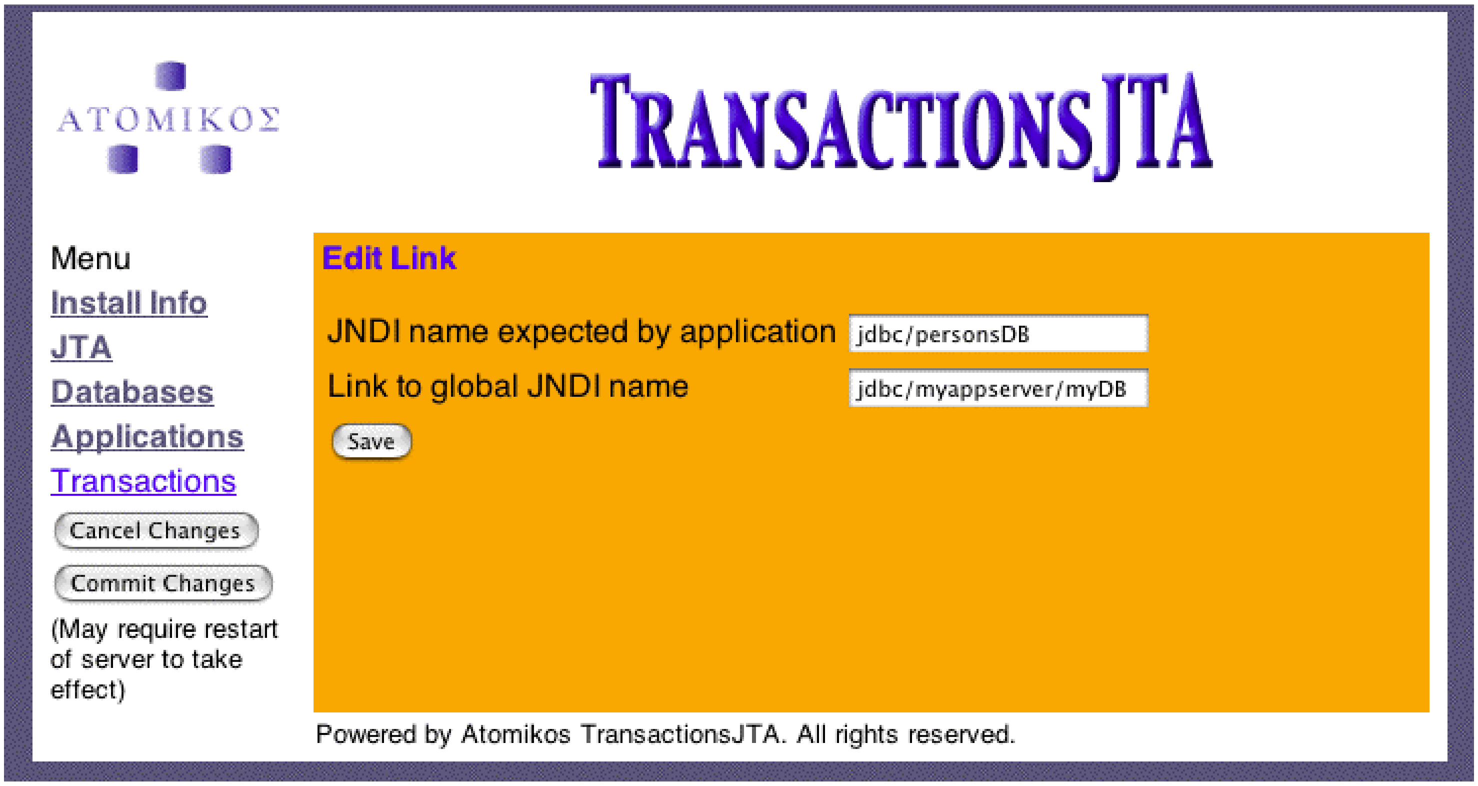
Powered by Atomikos TransactionsJTA. All rights reserved.

Click the button 'Add Resource Link' to install a new link for the application.

Caution

You need to define a link for each jdbc datasource declared in the application's deployment descriptor (the web.xml file in the .war archive for the application).

A screen appears with two values you need to supply:



JNDI name expected by application	jdbc/personsDB
Link to global JNDI name	jdbc/myappserver/myDB

JNDI name expected by application

> The JNDI name as declared in the application's deployment descriptor. This name is looked up by the application during its operation, and if the link is not defined here then the application will not be able to use the Atomikos database connector it needs.

Link to global JNDI name

> The global JNDI name of the Atomikos database connector that the application should use. By supplying this value, you effectively enforce the application to use this connector for its database access. Make sure to choose a JNDI name that belongs to a previously configured database connector (create the connector first if required).

Verifying JNDI Configuration

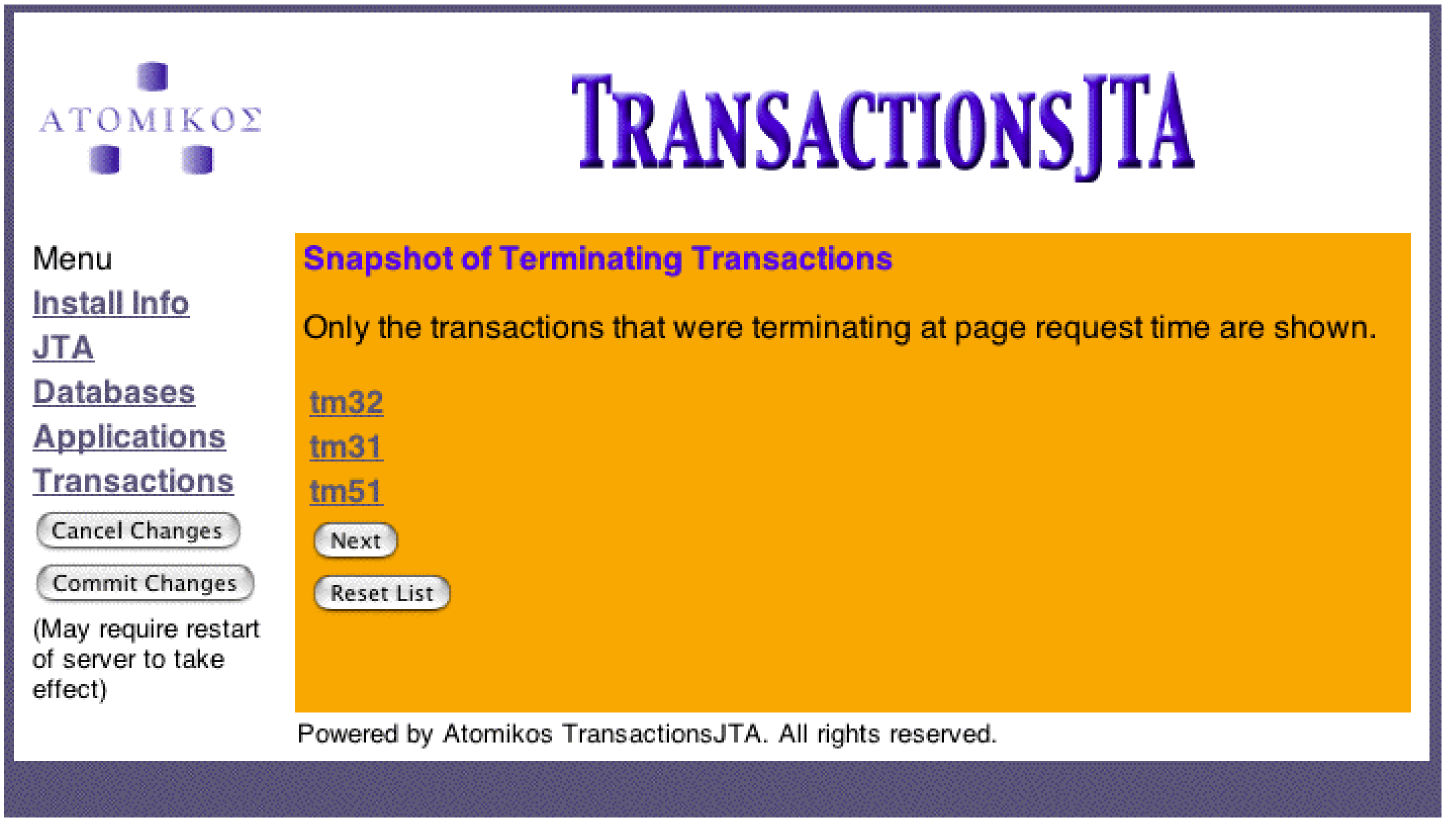
This concludes the application's configuration for TransactionsJTA™. When you click on 'Applications' in the menu, you should see the overview page of all applications and a status message for each of them. Your configured application should have a checkmark indicating that everything is properly setup (as far as can be detected). Once you are done you should commit your changes and restart the server.

Caution

The server has to be restarted before the applications will see the changes in your configuration's settings.

Chapter 4. Monitoring Ongoing Transactions

The menu option 'Transactions' shows a screen with all currently terminating transactions. Active transactions (those that are not terminating yet) are omitted, since the interesting cases are those that are in the termination phase. Usually this screen will be relatively empty, but if you repeatedly see the same transaction showing up then this could indicate a problematic transaction.



Snapshot of Terminating Transactions

Only the transactions that were terminating at page request time are shown.

tm32
tm31
tm51

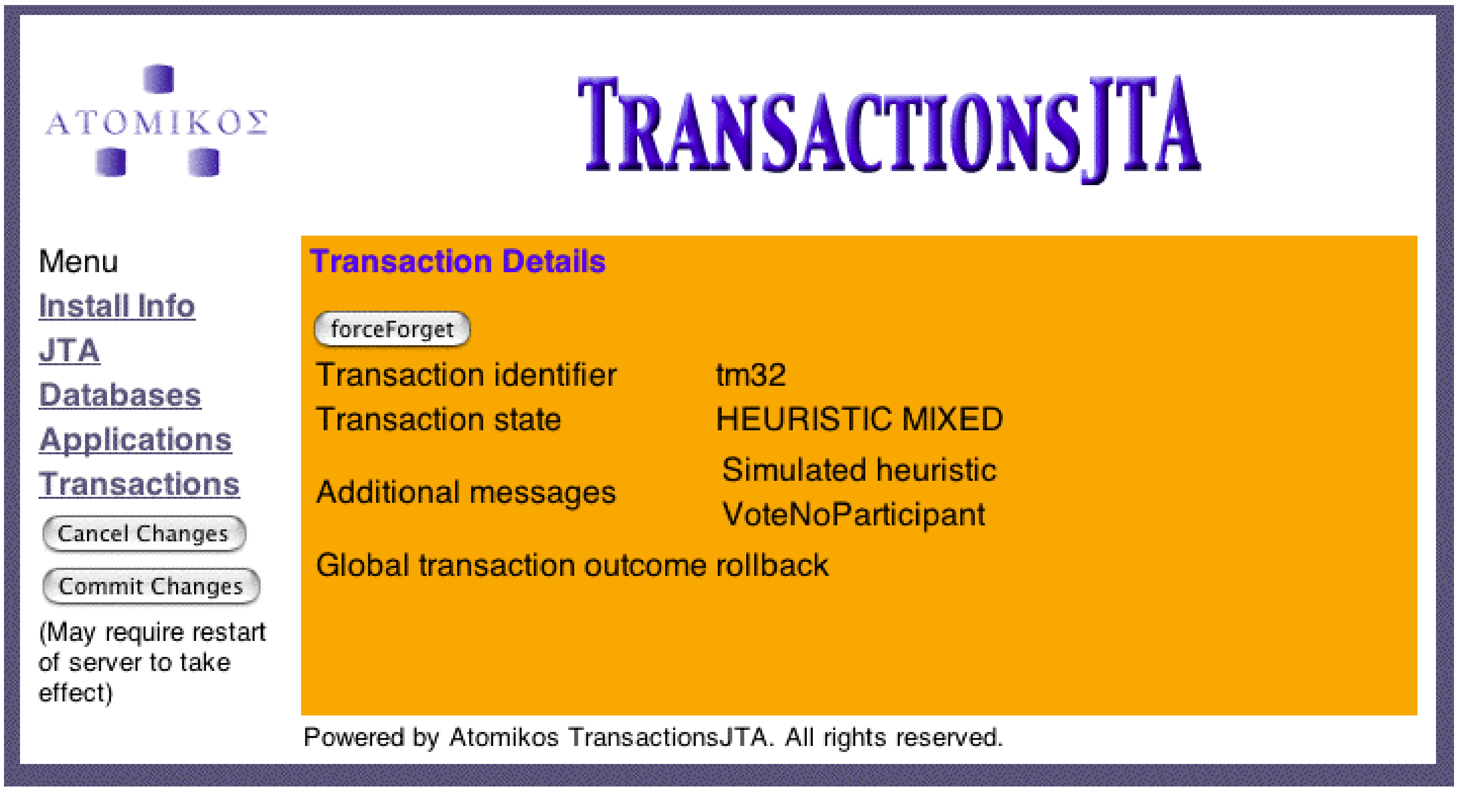
(May require restart of server to take effect)

Powered by Atomikos TransactionsJTA. All rights reserved.

Clicking on a transaction will show the transaction details as recorded in the logs (recording application-specific details in the logs to help in problem resolution is an Atomikos patent-pending technology). If you need to, you can force problematic transactions to be terminated and to be forgotten from the logs.

Caution

Forcing a transaction to terminate cannot be undone.



Transaction identifier	tm32
Transaction state	HEURISTIC MIXED
Additional messages	Simulated heuristic VoteNoParticipant
Global transaction outcome rollback	

Powered by Atomikos TransactionsJTA. All rights reserved.

Chapter 5. Programming Transactional Applications

Table of Contents

This chapter discusses some guidelines for developing transactional J2EE™ (web) applications. In general, TransactionsJTA™ follows the J2EE™ standards for transactions. This means that you can start and end transactions through an instance of javax.transaction.UserTransaction.

Starting a Transaction

Transactions are started by performing the following steps:

1. Lookup an instance of javax.transaction.UserTransaction in JNDI:

```
javax.naming.Context ctx = new InitialContext();
javax.transaction.UserTransaction utx =
     ctx.lookup ( "java:comp/env/UserTransaction" );
```

2. Call begin() on the UserTransaction instance:

   ```
   utx.begin();
   ```

Caution

Notice the /env/ subcontext for the lookup of the UserTransaction. This is Tomcat-specific and will be different on most other application servers. Please see the section on ensuring portability for more comments on how to make sure that your application will also work on other application servers.

Doing Transactional Work

Every database access that an application performs will be part of the active transaction, provided that the application uses a javax.sql.DataSource which is linked to an Atomikos database connector. See Create Required Links for details on how to link J2EE™ applications' datasources to Atomikos connectors.

Ending a Transaction

Ending the transaction is done by calling rollback or commit on the UserTransaction. However, to ensure that transactions are ALWAYS ended (and avoid pending transactions for threads) it is highly recommended that you perform this in a finally-block. A typical code pattern for transactions therefore looks as follows:

```
javax.naming.Context ctx = new InitialContext();
//lookup the transaction, NOTE the /env/ subcontext!
javax.transaction.UserTransaction utx =
     ctx.lookup ( "java:comp/env/UserTransaction" );
boolean commit = false;
try {
     utx.begin();

     //do the work that should be part of the transaction
     ...

     //if we reach this point the all went OK so we can commit
     commit = true;
}
catch ( Exception e ) {
     //if any exception happens, be prudent and do NOT commit
     commit = false;
     throw e;
}
finally {
     //terminate the transaction according to the outcome
     if ( commit ) utx.commit();
     else utx.rollback();
}
```

Ensuring Portability with other J2EE™ Servers

To ensure that the J2EE™ applications you write are portable to other application servers, your applications should start and end transactions as outlined in the previous sections. In addition, most application servers will make the UserTransaction available under a different name in JNDI (the J2EE™ specs recommend java:comp/ UserTransaction but Tomcat doesn't follow this guideline). For this reason, your web application should not hard-code the name in the source code but instead use a name whose value is declared as a JNDI environment parameter declared in the web.xml deployment descriptor (for instance, a context parameter). An example is shown below.

```
//the source code in your web application for looking up the UserTransaction

Context ctx = new InitialContext();
//the name to use for lookup is itself a String parameter in JNDI
String utxName = ( String ) ctx.lookup ( "UserTransactionName" );
UserTransaction utx = ( UserTransaction ) ctx.lookup ( utxName );
...
```

The corresponding web.xml deployment descriptor should then contain the adequate declaration for this to work:

```
<env-entry>
     <env-entry-name>UserTransactionName</env-entry-name>
     <env-entry-value>java:comp/env/UserTransaction</env-entry-value>
     <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

Note that if the application is installed on a different application server (with a different name for the UserTransaction) then all that needs to change is the value of this declaration in the web.xml file. No recompilation is needed.

Declarative Transaction Demarcation in Web Applications

For web applications, it is possible to use an URL-based declarative transaction demarcation scheme. With this strategy, transactions are started and ended outside the program logic, through the use of a J2EE™ filter class. The file JTA_INSTALLATION_FOLDER/samples/filters/TransactionFilter.java is an example of this technique. This filter needs to be installed on top of any URL that points to a transactional
task. Future releases of TransactionsJTA™ will include a predefined filter but in this release you still have to develop your own. The sample code can serve as an example to help you in this effort.

Using the Atomikos Transaction Tag

Another possibility for starting transactions in a JSP™ page is by means of the Atomikos transaction tag. This approach works as follows:

```
<!-- sample JSP page fragment -->
<!-- import the taglib and assign it a prefix jta -->
<%@ taglib uri="/WEB-INF/transactions.tld" prefix="jta" %>
<html>
...
<body>

<!-- start a transaction with a given timeout in seconds -->

<jta:transaction timeout ="30">

<!-- everything in here is part of the transaction -->
...


</jta:transaction>
<!-- the transaction is committed iff there was no exception -->

...

</body>
</html>
```

NOTE: the transaction tag looks up the UserTransaction in a context parameter, meaning that any web application that uses this tag needs to define a context parameter as follows:

```
<context-param>
     <param-name>javax.transaction.UserTransaction</param-name>
     <param-value>java:comp/env/UserTransaction</param-value>
</context-param>
```

Appendix A. Troubleshooting

Below is a list of common problems and their solutions. In general, when something doesn't work there should be more information in the command window or in the file catalina.log located in the logs folder in the tomcat directory.

Startup failed without any message.

Make sure that you have setup your environment as outlined in this guide. If not, please follow the setup steps. If you did setup your system correctly, then you can find out additional information as follows:

1. On Windows, open a command window and make sure you are in the bin folder of tomcat. Then, type 'catalina run' to see the messages that show up on the screen. The reason for startup failure should appear there. Contact support@atomikos.com if you need help.
2. On Linux/UNIX and MacOSX, check the terminal output and also the file 'catalina.out' in the logs folder. Use any information to solve the problem or contact support@atomikos.com for help.

Startup failed with 'NoSuchMethorError'.

This happens if you are using the wrong Java version. Make sure you are using Java 1.4 or higher.

'Commit Changes' Fails after Enabling an Application for TransactionsJTA™

This happens if an application's web.xml file has declared a resource-env-ref reference to the UserTransaction in JNDI. Suggested action: remove the reference from the web.xml file.

.zip Class Archives Not Loaded

Libraries installed in common/lib (such as JDBC™ drivers) are not loaded if they have a '.zip' extension. Make sure the driver's file name ends with '.jar' to avoid this problem.

I linked my application to an existing Atomikos database connector, but the application doesn't work any more

Please make sure that your application only accesses the database from within the scope of a transaction (use a transaction-starting filter web component if you can't change the source code). Atomikos' connectors require a transaction to work.

An XADataSource-specific field cannot be set to an empty value

The database connector edit screen does not (yet) allow you to clear an XADataSource-specific parameter after you saved it with a non-empty value. To circumvent this problem, delete and recreate the connector.

The JTA is not available

This is most likely the occur if you added TransactionsJTA to your existing Tomcat server (as opposed to using the preconfigured server pack). In that case, try the following:

1. Make sure that all required jars are in the common/lib folder
2. Make sure that you installed the atomikos.war application
3. In the atomikos application, visit 'Install Info' and press 'Commit Changes'
4. Make sure that you enabled your application for JTA (in the atomikos application, click on 'Applications')
5. Make sure that your application is referencing the UserTransaction under the right name. For Tomcat releases up to 5.0.18, this should be 'java:comp/env/UserTransaction'. For higher release numbers, use 'java:comp/UserTransaction'.
6. Restart the server.

If the problem still persists, please consult the catalina.out and the various *.txt files that are in the Tomcat log folder for more information. Contact support@atomikos.com if required.

Login Fails Repeatedly

If you can't login to the Atomikos control panel, please try the following:

1. Assert that the admin role is defined in tomcat-users.xml.
2. Assert that your user account is associated with that role.
3. Restart the server.
4. If all of the above are fine, then try again after restarting your browser.

The Atomikos Control Panel is Unavailable After Commit Changes

After you commit your changes, Tomcat tends to restart the web applications (including the control panel). For that reason, the control panel may become unavailable for a very short time after you commit your changes. Waiting until it has restarted should be sufficient to solve this.

Appendix B. References

Apache Tomcat

http://jakarta.apache.org/tomcat/

Atomikos

http://www.atomikos.com

FirstSQL™

http://www.firstsql.com

J2EE™ Web Applications: JSP™ Technology

http://java.sun.com/products/jsp

J2EE™ Web Applications: Servlet Technology

http://java.sun.com/products/servlet

The Online J2EE™ Tutorial

http://java.sun.com/j2ee/tutorial/1_3-fcs/

The Online JDBC™ Tutorial

http://java.sun.com/docs/books/tutorial/jdbc/

Appendix C. Database-Specific Connector Configuration Examples

Table of Contents

As long as your database has an XA-compliant JDBC connector available, it is possible to connect to it with TransactionsJTA and from your web applications. This is independent of the operating system, database vendor or platform on which your database server is installed. In general, you will find the setup details in the documentation of your JDBC product. This appendix shows how to connect to some common database products. Only the vendor-specific settings are shown.

FirstSQL 2.70 Enterprise Server

This section shows how to connect to FirstSQL with the default login created during the default installation process.

Full name of XADataSource class

Set this to: COM.FirstSQL.Dbcp.DbcpXADataSource

user

demo

password

You can leave this empty

portNumber

The port number on which the server is listening. The default installation uses 8000 (unless you changed it).

MS SQLServer

To connect to SQLServer™ with transactional guarantees, you need to use XA-capable JDBC drivers. One place to get such drivers is at DataDirect (http://www.datadirect.com). In the rest of this section, we assume that you are using the DataDirect Connect for JDBC™ drivers. The following list explains the steps to connect to your SQLServer:

1. Copy DataDirect's required jar files (base.jar, util.jar and sqlserver.jar) into Tomcat's common/lib folder. Restart Tomcat to make sure that the files are loaded by the server.
2. Make sure that the Distributed Transaction Coordinator™ is running on your SQLServer machine. To check, go to the 'My Computer' icon, right-click and choose Manage->Services to see all installed services. Make sure that the required service is running.
3. Make sure that your SQLServer is configured for authentication: open the SQLServer Enterprise Manager application, right-click your SQLServer and choose properties from the pop-up menu. Click the Security tab, and make sure to select 'SQL Server and Windows' for the 'Authentication' setting.
4. Create a new database connector in the Atomikos control panel application, and make sure to set the following specific parameters:

    Full name of XADataSource class

        Set this to: com.ddtek.jdbcx.sqlserver.SQLServerDataSource

    user

        Your SQLServer username.

    password

Your SQLServer password.

serverName

Set this to the IP address of your SQLServer machine. If you have more than one (default) SQLServer instance running on that machine, you may also have to specify the instance name (please check the DataDirect/SQLServer documentation on how to do that).

databaseName

Set this to the name of the database (managed by the server instance) that you want to connect to.

Oracle (8.1.7 or higher)

Oracle™ is rather special in its configuration, and you will not be able to do transactions correctly unless you follow these guidelines.

Format for XID instances

The value of this field should be 'Oracle'.

Don't reuse connections before commit

This should be checked.

Full name of XADataSource class

Set this field to 'oracle.jdbc.xa.client.OracleXADataSource'.

Validating query

SELECT SYSDATE FROM DUAL

URL

Simply set this value to jdbc:oracle:thin:USERNAME/PASSWORD@HOSTNAME:PORT:SID where the upper-case names are to be replaced with the values appropriate for your server installation. Please consult the Oracle documentation for more details. NOTE: if you set this XA-specific property then there is no need to set the other (Oracle-specific) XA properties. This URL should be enough in most cases.

Caution

Before you can start JTA/XA transactions in Oracle, your server needs to be configured for XA. On most versions, this is done by running two administrative scripts: 'initxa.sql' and 'xaview.sql'. You can find these scripts in your Oracle installation folder. Alternatively, you can ask your database administrator to setup your server.

Caution

Oracle does not support SQL DDL statements such as creation or deletion of tables through transactional connectors. In other words, you cannot create or delete tables via a connection gotten through an Oracle XADataSource.

Caution

Make sure that the Oracle JDBC driver are copied in the common/lib folder of TransactionsJTA. This driver is included in your Oracle installation (look for classes12.zip). Make sure to rename the file to classes12.jar or it will not load correctly in Tomcat! Restart the server afterwards.

Appendix D. Using TransactionsJTA with Tomcat 4.x

Table of Contents

Although it is not recommended, technically it is feasible to run the software on Tomcat 4.x. The disadvantage is that the atomikos web application cannot be used to configure the server. Instead, you have to manually edit the server.xml file for Tomcat.

Installation

Add the jar files in the lib folder of TransactionsJTA to the common/lib folder of Tomcat. Also make sure the deploy the application 'atomikos.war' or transactions will not be available. See the normal installation instructions earlier in this manual for more information on this task.

Configure Tomcat

The configuration of Tomcat is all done in the server.xml file (located in tomcat's conf folder). A useful example is the pre-configured server.xml file that you find in this release (you should be able to use the same file on Tomcat 4.x). There are different steps to configuring Tomcat: adding global JNDI resources (UserTransaction and Atomikos database connectors) and configuring individual applications (by adding resource links).

Caution

Although you have to install the 'atomikos.war' application, this application will not be able to configure

Tomcat 4.x servers for you. Instead of adding connectors and setting up applications in this tool, you will have to do all setup manually by editing the file server.xml. For maximum setup comfort, we recommend that you use only Tomcat 5.0.18 or higher.

Enable Tomcat 4.x for TransactionsJTA

To enable Tomcat 4.x for this product, you need to insert a global naming resource element for the UserTransaction:

```
<Server>
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"/>
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
  <GlobalNamingResources>

    <Resource auth="Container" name="UserTransaction"
        type="com.atomikos.icatch.jta.tomcat.UserTransactionImp"/>

    <ResourceParams name="UserTransaction">
      <parameter>
        <name>factory</name>
        <value>org.apache.naming.factory.BeanFactory</value>
      </parameter>
    </ResourceParams>
    ...
   </GlobalNamingResources>
   ...
</Server>
```

Insert Atomikos Database Connectors

In addition, you will need a database connector AND an XADataSource entry for each database you want to access. The following shows how this is done for the pre-configured demo database connectors:

```
<Server>
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"/>
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
  <GlobalNamingResources>
    ...
    <!--
        An Atomikos DB connector: choose your own name,
        but make sure that the type is the same
    -->
    <Resource auth="Container" name="jdbc/com/atomikos/demo/personsDB"
        type="com.atomikos.jdbc.DataSourceBean"/>

    <!-- The corresponding XA connector: choose your own name,
        and change to type to your JDBC vendor's
    -->
    <Resource name="jdbc/xa/com/atomikos/demo/personsDatabase"
        type="COM.FirstSQL.Dbcp.DbcpXADataSource"/>

    <!-- The properties for the personsDB connector:
          notice that uniqueResourceName must be equal to the name
          and dataSourceName must be the name of the XA connector or things will not work.
          In any case, add the same factory parameter since Tomcat needs it.
    -->
    <ResourceParams name="jdbc/com/atomikos/demo/personsDB">
      <parameter>
        <name>xidFormat</name>
        <value>Default</value>
      </parameter>
      <parameter>
        <name>connectionPoolSize</name>
        <value>1</value>
      </parameter>
      <parameter>
        <name>dataSourceName</name>
        <value>jdbc/xa/com/atomikos/demo/personsDatabase</value>
      </parameter>
      <parameter>
        <name>factory</name>
        <value>org.apache.naming.factory.BeanFactory</value>
      </parameter>
      <parameter>
        <name>validatingQuery</name>
        <value>select * from persons</value>
      </parameter>
      <parameter>
        <name>exclusiveConnectionMode</name>
        <value>false</value>
      </parameter>
      <parameter>
        <name>uniqueResourceName</name>
        <value>jdbc/com/atomikos/demo/personsDB</value>
      </parameter>
      <parameter>
        <name>connectionTimeout</name>
        <value>15</value>
      </parameter>
      <parameter>
        <name>statementTimeout</name>
        <value>1</value>
      </parameter>
    </ResourceParams>

    <!-- The properties for the XADataSource, add one for each vendor-specific property
        you want to set. In any case, add the same factory parameter since Tomcat needs it.
    -->
    <ResourceParams name="jdbc/xa/com/atomikos/demo/personsDatabase">
      <parameter>
        <name>driverPackage</name>
        <value>COM.FirstSQL.Dbcp</value>
      </parameter>
      <parameter>
        <name>factory</name>
        <value>org.apache.naming.factory.BeanFactory</value>
      </parameter>
      <parameter>
        <name>user</name>
        <value>demo</value>
      </parameter>
      <parameter>
        <name>networkProtocol</name>
        <value>dbcp</value>
      </parameter>
      <parameter>
```

```
        <name>portNumber</name>
        <value>8000</value>
      </parameter>
      <parameter>
        <name>loginTimeout</name>
        <value>0</value>
      </parameter>
    </ResourceParams>
    ...
   </GlobalNamingResources>
   ...
</Server>
```

Configure Each Application

Every J2EE application that you want to enable for transactions must be configured with links. Below is the configuration to make the egov demo application use the UserTransaction and database connectors we installed in the above:

```
<Server>

  <GlobalNamingResources>
    ...
   </GlobalNamingResources>
   ...

   <Context path="/egov" docBase="egov">

                 <!-- This link makes UserTransaction available to this application;
                     copy this without changes
                -->
                 <ResourceLink global="UserTransaction"
                     name="UserTransaction"
                      type="com.atomikos.icatch.jta.tomcat.UserTransactionImp"/>

                 <!-- This link makes the Atomikos db connector available under the given name
                      and corresponds to the connector with the global name.
                      Change 'jdbc/personsDB' to the name your application needs and
                      change 'jdbc/com/atomikos/demo/personsDB' to your
                      connector's global resource name.
                -->
                 <ResourceLink global="jdbc/com/atomikos/demo/personsDB"
                     name="jdbc/personsDB"
                      type="com.atomikos.jdbc.DataSourceBean"/>

                 <!-- This link makes the XADataSource available.
                       Make sure that BOTH name and global have the SAME value
                       as the global resource name of your XADataSource.
                       Change the type to your JDBC vendor's XADataSource class name.
                -->
                  <ResourceLink global="jdbc/xa/com/atomikos/demo/personsDatabase"
                      name="jdbc/xa/com/atomikos/demo/personsDatabase"
                      type="COM.FirstSQL.Dbcp.DbcpXADataSource" />

            </Context>
```

```
</Server>
```

Appendix E. Using TransactionsJTA with Tomcat 5.0.19 and later

Table of Contents

Installing TransactionsJTA

Add the jar files in the lib folder of TransactionsJTA to the common/lib folder of Tomcat. Also make sure the deploy the application 'atomikos.war' or transactions will not be available. See the normal installation instructions earlier in this manual for more information on this task.

UserTransaction in JNDI

In the Tomcat 5.0.19 release the UserTransaction anomaly has been fixed. This means that this release of Tomcat follows the normal J2EE convention and makes the UserTransaction available under the name java:comp/UserTransaction.

Running the Egov Demo

To run the Egov demo, follow these steps:

1. Install the demo application
2. Setup 2 Atomikos database connectors for FirstSQL
3. Configure the Egov application for JTA
4. Add 2 links: one for jdbc/personsDB and another for jdbc/marriagesDB
5. Commit Changes
6. Open the admin application under the URL /admin
7. Login with your admin account
8. Go to the egov application context
9. Edit the environment entries: change the value of the parameter "UserTransactionName" to java:comp/UserTransaction (instead of java:comp/env/UserTransaction)
10. Save your changes
11. Open the manager application on the URL /manager/html

12. Login with your manager account
13. Reload the egov context

Running the testJTA Demo

To run the testJTA application in Tomcat 5.0.19 or higher, you first need to edit the web.xml file and change the context parameters as follows:

```
<context-param>
       <param-name>javax.transaction.UserTransaction</param-name>
       <param-value>java:comp/UserTransaction</param-value>
</context-param>
```

Then (re)start the application.

Appendix F. Integration with Eclipse™ and IBM Websphere™

The folder 'extra/IBM' contains a plugin that allows you to integrate Tomcat with these products. Because this release of TransactionsJTA runs inside Tomcat, you can also use this plugin to integrate TransactionsJTA in these same products. To install this plugin, unzip it in the eclipse/plugins folder and restart your workbench. Then follow the instructions in the included documentation.