Configuration

Configuration

BTM configuration settings are stored in a `Configuration` object. It can be obtained by calling `TransactionManagerServices.getConfiguration()`. All settings are documented in the javadoc and you should refer to it to know what can be configured.

Contents

- How to configure BTM
  - The properties configuration file
  - Setting values directly on the `Configuration` object
- Configurable settings
  - Transaction engine settings
  - Journal settings
  - Timers settings
  - Resource Loader settings
- Connection pools settings

How to configure BTM

The `Configuration` object is implemented with sensible default settings. For a first time user, all default settings are good enough. After the initial testing phase, you might want to change some settings. This can be done in two ways: via a properties configuration file or by setting values directly on the `Configuration` object.

The properties configuration file

You can create a properties file in which you'll set some configuration settings. All the ones you omit will keep their default value.

The file can be stored anywhere on the file system in which case you need to set the `bitronix.tm.configuration` system property to tell BTM where the file lies. This is generally done by adding a `-D` argument to the virtual machine's command line:

```
java
-Dbitronix.tm.configuration=./my-btm-config.
properties MyClass
```

Another way is to call the properties file `bitronix-default-config.properties` and store it at the root of your classpath.

The properties file is in the default format `key=value`. Ant-like references (`${some.property.name}`) to other properties or to system properties (defined with `-D` on the command line) are supported.

Setting values directly on the `Configuration` object

You can call any setter you want on the object you get from the call to `TransactionManagerServices.getConfiguration()`. This is convenient if you do not want to use the properties file to configure BTM but want to leverage - *for instance* - Spring instead.

```
Configuration conf =
TransactionManagerServices.getConfiguration(
);
conf.setServerId("jvm-1");
conf.setLogPart1Filename("./tx-logs/part1.bt
m");
conf.setLogPart2Filename("./tx-logs/part2.bt
m");
```

> **Read only configuration**
>
> Once BTM has started, any attempt to call a set method on the `Configuration` object will throw a `IllegalStateException`.

Since the `Configuration` object is a singleton, there is no need to pass it to any other object, BTM will pick it up at startup.

> **Configuration dropped after restart**
>
> The `Configuration` object is dropped during BTM shutdown. You need to call all setters methods again before you restart the transaction manager. Keep this in mind if you plan to hot-redeploy your application in your application server.

Configurable settings

There are many different settings that are configurable in the transaction manager. Fortunately, all defaults settings are usually good enough to get started. You only need to tune them when required.

Transaction engine settings

These configurable properties are related to the transaction manager's core.

File property name	Configuration property name	Default value	Description

bitronix.tm.serverId	serverId	The machine's IP address but that's unsafe for production usage	a stable ASCII string that must uniquely identify this TM instance. It must not exceed 51 characters or it will be truncated.
bitronix.tm.2pc.async	asynchronous2Pc	false	Should two phase commit be executed asynchronously? Asynchronous two phase commit will improve 2PC execution time when there are many resources enlisted in transactions but can be very CPU intensive when used on JDK 1.4 without the java.util.concurrent backport implementation available on the classpath. It also makes debugging more complex. See here for more details.
bitronix.tm.2pc.warnAboutZeroResourceTransactions	warnAboutZeroResourceTransaction	true	Should transactions executed without a single enlisted resource result in a warning or not? Most of the time transactions executed with no enlisted resource reflect a bug or a mis-configuration somewhere.
bitronix.tm.2pc.debugZeroResourceTransactions	debugZeroResourceTransaction	false	Should creation and commit call stacks of transactions executed without a single enlisted resource tracked and logged or not? This is a companion to warnAboutZeroResourceTransaction where the transaction creation and commit call stacks could help you identify the culprit code.

bitronix.tm.disableJmx	disableJmx	false	The transaction manager registers objects in the JMX registry by default if available. Set this to true to never register JMX objects.
bitronix.tm.jndi.userTransactionName	jndiUserTransactionName	java:comp/UserTransaction	The name under which the transaction manager will be bound in the internal JNDI provider.
bitronix.tm.allowMultipleLrc	allowMultipleLrc	false	Should the transaction manager allow multiple LRC resources to be enlisted into the same transaction? Having multiple LRC resources participate in a transaction gives up the recovery guarantee but sometimes is useful in development mode.
bitronix.tm.currentNodeOnlyRecovery	currentNodeOnlyRecovery	true	Set this to true if you run multiple instances of the transaction manager on the same JMS and JDBC resources to avoid the recovery process to try to recover transactions started by another node. See here for more details.

Journal settings

These configurable properties are related to the disk journal used to record recovery information.

File property name	Configuration property name	Default value	Description

bitronix.tm.journal	journal	disk	Set the journal to be used to record transaction logs. This can be any of **disk**, **null** or a class name. The disk journal is a classic implementation using two fixed-size files and disk forces, the null journal just allows one to disable logging. This can be useful to run tests. ***Do not use the null journal on production as without transaction logs, atomicity cannot be guaranteed.***
bitronix.tm.journal.disk.logPart1Filename	logPart1Filename	btm1.tlog	Journal fragment file 1.
bitronix.tm.journal.disk.logPart2Filename	logPart2Filename	btm2.tlog	Journal fragment file 2.
bitronix.tm.journal.disk.forcedWriteEnabled	forcedWriteEnabled	true	Are logs forced to disk? ***Do not set to false on production since without disk force, atomicity cannot be guaranteed.***
bitronix.tm.journal.disk.forceBatchingEnabled	forceBatchingEnabled	true	Are disk forces batched? Disabling batching can seriously lower the transaction manager's throughput.
bitronix.tm.journal.disk.maxLogSize	maxLogSize	2	Maximum size in megabytes of the journal fragments. Larger logs allow transactions to stay longer in-doubt but the TM pauses longer when a fragment is full.
bitronix.tm.journal.disk.filterLogStatus	filterLogStatus	false	Should only mandatory logs be written? Enabling this parameter lowers space usage of the fragments but makes debugging more complex.

bitronix.tm.journal.disk.skipCorruptedLogs	skipCorruptedLogs	false	Should corrupted transactions log entries be skipped? Use only at last resort when all you have to recover is a pair of corrupted files.

Timers settings

The transaction manager heavily relies on timeouts. All of them can be configured.

File property name	Configuration property name	Default value	Description
bitronix.tm.timer.defaultTransactionTimeout	defaultTransactionTimeout	60	Default transaction timeout in seconds.
bitronix.tm.timer.gracefulShutdownInterval	gracefulShutdownInterval	60	Maximum amount of seconds the TM will wait for transactions to get done before aborting them at shutdown time.
bitronix.tm.timer.backgroundRecoveryInterval	backgroundRecoveryInterval	0	**Deprecated** Interval in minutes at which to run the recovery process in the background. Cannot be disabled.
bitronix.tm.timer.backgroundRecoveryIntervalSeconds	backgroundRecoveryIntervalSeconds	60	Interval in seconds at which to run the recovery process in the background. Cannot be disabled.

Resource Loader settings

The resource loader loads and configures XA resources using configuration stored in a properties file. See the Resource Loader page for more details.

File property name	Configuration property name	Default value	Description
bitronix.tm.resource.configuration	resourceConfigurationFilename	none *(optional)*	Resource Loader configuration file name.

Connection pools settings

JDBC and JMS connection pools configuration are discussed in details in the JDBC pools configuration and the JMS pools configuration pages. Alternatively you can use the Resource Loader instead.