DebugLogging13

Debug logging

BTM ships with SLF4J which is a logging facade much like Apache Commons Logging but it's safe to use in application servers unlike Commons Logging which suffers from complex classloader issues.

Much like Commons Logging, SLF4J can route the log requests to many different implementations including the two most common ones: JDK 1.4 and Log4J.

By default, BTM is shipped with slf4j-jdk14-1.5.8.jar which is the the JSR 047 logging API (aka JDK 1.4 logging) wrapper. If you want to use another logger, simply delete the slf4j-jdk14-1.5.8.jar file and replace it with any one from this list:

- slf4j-jdk14-1.5.8.jar for JSR 047 logging.
- slf4j-log4j12-1.5.8.jar for Apache Log4J 1.2.x.
- slf4j-nop-1.5.8.jar for no logging (logs are silently dropped).
- slf4j-simple-1.5.8.jar for simple `System.out` output.

You will also need to put the slf4j-api-1.5.8.jar file in your classpath as well, no matter which one of the above jars you choose.

Those jars come from the 1.5.8 distribution of SLF4J.

Mapped Diagnostic Context

This feature appeared in BTM 1.3.3 and is not available in earlier versions.

If your logging implementation supports MDC (like Log4J does for instance) BTM will push the transaction's GTRID in the logger's MDC when a transaction is created.

The GTRID is pushed under the **`btm-gtrid`** name. You can make it appear in your Log4J logs by adding it to the `ConversionPattern`:

```
<appender name="FileAppender"
class="org.apache.log4j.FileAppender">
    <param name="File"
value="SomeLogFile.txt"/>
    <layout
class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
value="%X{btm-gtrid} - %m%n"/>
    </layout>
 </appender>
```

When using a different implementation than Log4J, please refer to your logger's documentation to understand how to make use of this feature.