Logging, Traces and Alarms

Table of Contents

Log4j Logging Service

In Restcomm JAIN SLEE `Apache log4j` is used for logging. If you are not familiar with the log4j package and would like to use it in your applications, you can read more about it at the Jakarta web site.

Logging is controlled from a central *conf/jboss-log4j.xml* file, in each server configuration profile. This file defines a set of appenders specifying the log files, what categories of messages should go there, the message format and the level of filtering. By default, in Restcomm produces output to both the console and a log file (*log/server.log*).

There are 6 basic log levels used: TRACE, DEBUG, INFO, WARN, ERROR and FATAL.

Logging is organized in categories and appenders. Appenders control destination of log entries. Different appenders differ in configuration, however each supports threshold. Threshold filters log entries based on their level. Threshold set to WARN will allow log entry to pass into appender if its level is WARN, ERROR or FATAL, other entries will be discarded. For more details on appender configuration please refer to its documentation or java doc.

The logging threshold on the console is INFO, by default. In contrast, there is no threshold set for the server.log file, so all generated logging messages are logged there.

Categories control level for loggers and its children, for details please refer to log4j manual.

By default Restcomm JAIN SLEE inherits level of INFO from root logger. To make platform add more detailed logs, file *conf/jboss-log4j.xml* has to be altered. Explicit category definition for Restcomm JAIN SLEE looks like:

```
<category name="org.mobicents.slee">
    <priority value="INFO"/>
</category>
```

This limits the level of logging to INFO for all Restcomm JAIN SLEE classes. It is possible to declare more categories with different level, to provide logs with greater detail.

For instance, to provide detailed information on Restcomm JAIN SLEE transaction engine in separate log file(*txmanager.log*), file *conf/jboss-log4j.xml* should contain entries as follows:

```
<appender name="TXMANAGER" class="org.jboss.logging.appender.RollingFileAppender">
    <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
    <param name="File" value="${jboss.server.home.dir}/log/txmanager.log"/>
    <param name="Append" value="false"/>
    <param name="MaxFileSize" value="500KB"/>
    <param name="MaxBackupIndex" value="1"/>

    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
    </layout>
</appender>

<category name="org.mobicents.slee.runtime.transaction">
    <priority value="DEBUG" />
    <appender-ref ref="TXMANAGER"/>
</category>
```

This creates a new file appender and specifies that it should be used by the logger (or category) for the package org.mobicents.slee.runtime.transaction.

The file appender is set up to produce a new log file every day rather than producing a new one every time you restart the server or writing to a single file indefinitely. The current log file is txmanager.log. Older files have the date they were written added to their filenames.

Simplified Global Log4j Configuration

Besides manual logging configuration, described previously, Restcomm JAIN SLEE also exposes management operations that greatly simplify such configuration, allowing the administrator to select through predefined and complete logging configuration presets. Such operations are available in MBean named *org.mobicents.slee%3Aservice%3DMobicentsManagement*, and the available presets are:

Level

The available management operations are:

- DEFAULT: Regular logging, at INFO level, displaying most user-related messages;
- DEBUG: More verbose logging, mostly using DEBUG/TRACE level, displaying message of interest for developers;
- PRODUCTION: Low verbosity and async logging, mostly in WARN level, for systems in production so that logging does impact performance.

JMX Operation

- getLoggingConfiguration: retrieves what is the current logging configuration;
- switchLoggingConfiguration: allows switching to a different configuration preset;
- setLoggingConfiguration: used to upload a complete logging configuration.

Custom presets can be easily deployed in the application server too. Simply name the configuration file as *jboss-log4j.xml.PRESET_NAME*, where `PRESET_NAME` should be unique preset name, and copy it to directory *$JBOSS_HOME/server/profile_name/deploy/restcomm-slee/log4j-templates*, where `profile_name` is the server profile name.

These procedures changes the whole JBoss Application Server Platform logging configuration, so it will affect also logging for other running applications besides the JAIN SLEE container.

Alarm Facility

The `JAIN SLEE Alarm Facility` is used by SBBs, Resource Adaptors, and Profiles to request the SLEE to raise or clear alarms. If a request is made to raise an alarm and the identified alarm has not already been raised, the alarm is raised and a corresponding alarm notification is generated by the `AlarmMBean`. If a request is made to clear an alarm and the identified alarm is currently raised, the alarm is cleared and a corresponding alarm notification is generated by the `AlarmMBean`.

Alarm notifications are intended for consumption by management clients external to the SLEE. The management client is responsible for registering to receive alarm notifications generated by the Alarm Facility through the external management interface of the `Alarm Facility`. The management client may optionally provide notification filters so that only the alarm notifications that the management client would like to receive are transmitted to the management client.

For further information on how to use `JAIN SLEE Alarm Facility` and receive JMX notifications refer to the JAIN SLEE 1.1 Specification.

Trace Facility

Notification sources such as SBBs, Resource Adaptors, Profiles, and SLEE internal components can use the `Trace Facility` to generate trace messages intended for consumption by external management clients. Management clients register to receive trace messages generated by the `Trace Facility` through the external management interface (MBean). Filters can be applied, in a similar way as in case of Alarms.

Within the SLEE, notification sources use a `tracer` to emit trace messages. A tracer is a named entity. Tracer names are case-sensitive and follow the Java hierarchical naming conventions. A tracer is considered to be an ancestor of another tracer if its name followed by a dot is a prefix of the descendant tracer's name. A tracer is considered to be a parent of a tracer if there are no ancestors between itself and the descendant tracer. For example, the tracer named `com` is the parent tracer of the tracer named `com.foo` and an ancestor of the tracer named `com.foo.bar`.

All tracers are implicitly associated with a notification source, which identifies the object in the SLEE that is emitting the trace message and is included in trace notifications generated by the `Trace MBean` on behalf of the tracer. For instance, an SBB notification source is composed by the SBB id and the Service id.

Multiple notification sources may have tracers with same name in SLEE. Comparing with common logging frameworks, this would mean that the notification source would be part of the log category or name.

For further information on how to use `JAIN SLEE Trace Facility` and receive JMX notifications refer to the JAIN SLEE 1.1 Specification.

JAIN SLEE Tracers and Log4j

Restcomm JAIN SLEE Tracers additionally log messages to `Apache Log4j`, being the log4j category, for notification source `X`, defined as `javax.slee.` concatenated with the `X.toString()`.

For instance, the full log4j logger `name` for tracer named `GoogleTalkBotSbb`, of sbb notification source with `SbbID[name=GoogleTalkBotSbb,vendor=restcomm,version=1.0]` and `ServiceID[name=GoogleTalkBotService,vendor=restcomm,version=1.0]`, would be `javax.slee.SbbNotification[service=ServiceID[name=GoogleTalkBotService, vendor=restcomm,version=0.1], sbb=SbbID[name=GoogleTalkBotSbb,vendor=restcomm, version=0.1]].GoogleTalkBotSbb` (without the spaces or breaks), which means a log4j category defining its level as `DEBUG` could be:

```
<category
    name="javax.slee.SbbNotification[service=ServiceID[name=GoogleTalkBotService,
    vendor=restcomm,version=0.1],sbb=SbbID[name=GoogleTalkBotSbb,
    vendor=restcomm,version=0.1]]">
    <priority value="DEBUG" />
</category>
```

The relation of JAIN SLEE `tracers` and log4j `loggers` goes beyond log4j showing tracer's messages, changing the tracer's log4j logger `effective level` changes the tracer level in SLEE, and vice-versa. Since JAIN SLEE tracer levels differ from log4j logger levels a mapping is needed:

Table 1. Mapping JAIN SLEE Tracer Levels with Apache Log4j Logger Levels

Tracer Level	Logger Level
OFF	OFF
SEVERE	ERROR
WARNING	WARN
INFO	INFO
CONFIG	INFO
FINE	DEBUG
FINER	DEBUG
FINEST	TRACE