# Log Service Update

Draft

9 Pages

*Text in Red is here to help you. Delete it when you have followed the instructions.*
*The <RFP Title> can be set from the File>Properties:User Defined menu. To update it onscreen, press F9. To update all of the fields in the document Select All (CTRL-A), then hit F9. Set the release level by selecting one from: Draft, Final Draft, Release.  The date is set automagically when the document is saved.*

## Abstract

Log Service has an important role in the software deployment. It can discover the reason for a blocker issue. The current deployments are using various log API like: Log4j 2, SLF4J/Logback, Equinox Log etc. The evolution of the log API calls for new Log Service requirements. This document contains proposal for updates to improve the Log Service usage.

# 0 Document Information

## 0.1 License

### DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable, worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future (“Future Specification”), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

## 0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

## 0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

## 0.4 Table of Contents

## 0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 6.1.

```
Source code is shown in this typeface.
```

## 0.6 Revision History

The last named individual in this history is currently responsible for this document.

| Revision | Date | Comments |
|---|---|---|
| Initial | Sep 17 2014 | Initial version introduced on F2F Meeting in Madrid, Spain hosted by Liferay, September 9-11, 2014.<br>Evgeni Grigorov, ProSyst Software, e.grigorov@prosyst.com |

# 1 Introduction

It's a great feature that OSGi Log Service is available in the OSGi service registry. Many providers can register their implementations at the same time. The services can be unregistered and registered to provides some updates. That dynamism is good for the providers, but complicates the applications logging.

When the logging is the next application step, the hard work begins. The application should track the Log Service from the OSGi service registry. If there are no Log Services, the application should buffer or print the log message. If there are more than one Log Services, the application should log the message through all of them.

Those complications can be cleared and single API can be introduced. The new interface can be used by the framework and all active bundles. There will be single log entry point.

# 2 Application Domain

Most of the applications are using the well-known logging API like Log4j 2 and SLF4J/Logback. They expect to have simple and static way to receive a logger. The applications request different loggers for their internal modules. The log messages are going to be more and more flexible with support of:

- Parametrization – the message can be filled with different parameters at runtime.
- Context – the message is automatically assigned to a context with additional details like thread info.

List of currently used logging frameworks:

- Log4j and the next version Log4j 2
- SLF4J and the update Logback
- Java 2 Logging API
- Equinox Log Service (org.eclipse.equinox.log)

## 2.1 Terminology + Abbreviations

- SLF4J - Simple Logging Facade for Java

# 3 Problem Description

The current OSGi approach about the logging facilities is convenient for the Log Service providers, but complicates the consumers. It can be improved to minimize the logging source code, to simplify the usage and to keep the application source code clean and readable.

The applications have an issue if the Log Service is missing. They need to buffer the log entries or print them on the console. If more than one Log Service is available, all of them have to be notified in case of new log entry. Debugging is all about the process to discover new defects and there is a need for as much as possible information. The log messages can be colored with additional details like thread info, log name, context etc.

# 4 Use Cases

## 4.1 Single OSGi Log

The OSGi framework integrator would like to have a single Log Service. It'll simplify the log configuration and will place all log messages in a single place. Currently, the bundles are using the Log Service, but the framework (system bundle) is using another own logger.

## 4.2 Log Service Availability

The OSGi framework is a dynamic environment. The services can be registered and unregistered over and over again. The Log Service is not an exception. The bundles must be prepared with a backup log option in case of missing Log Service. It complicates the implementation and often there are some Log Service wrappers to handle this scenario.

## 4.3 Log Message Parameterization

The bundle implementation has detailed log. The source code is not readable and hides the real flow because of a lot of:

```
if (isDebugEnabled) {

  logger.debug(“File:‟ + fileName + ‟ has been processed.”);

}
```

The source code will be much clear if this debug is replaced by:

```
logger.debug("File:{} has been processed.", fileName);
```

## 4.4 Named Log Service

The application bundle is internally organized in modules. There are: configuration, db and UI modules. Each module needs own logger to log the application flow. Currently, all modules are sharing the same Log Service and are using a message prefix to mark that the message is coming from the given module. An example:

```
logger.info("[DB] Database has been restored.")
```

It'll be convenient to request the logger with a name. The name can be mapped to the bundle internal module name. The prefix from the example will be redundant then.

## 4.5 Log Entry Enhancement

The system doesn't have an option for remote debug and there is only one option, activated debug. The debug messages must contain as much as possible details like: thread info, logger name, sequence number etc. Currently, those details are missing in OSGi LogEntry and custom solutions are used.

## 4.6 Multiple Log Services

The application bundle has a critical log message. The OSGi framework maintains a few Log Services. The application bundle should visit all Log Services to submit the message. That scenario can be simplified and automated.

# 5 Requirements

## 5.1 Application Requirements

1. The solution SHOULD provide an option for a single OSGi Log. Log can be used by the bundles and OSGi framework itself (system bundle).

2. The solution SHOULD guarantee the availability of the OSGi Log. In this way, the applications can have the assumption that there is always OSGi Log.

3. The solution SHOULD provide an option for parameterized log messages.

4. The solution SHOULD provide an option for name association with the logger i.e. named logger.

5. The solution MAY provide new log levels like: trace.

6. The solution MUST use OSGi whiteboard pattern for the Log Listener.

7. The solution SHOULD provide a mechanism for a notification of multiple Log Services.

## 5.2 Log Entry Requirements

1. The solution SHOULD provide the long entry with thread info.
2. The solution MAY provide location info like class, method etc.
3. The solution SHOULD provide the log entry with a sequence number.
4. The solution SHOULD provide the log entry with a logger name.

# 6 Document Support

## 6.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

[3]. https://issues.apache.org/jira/browse/FELIX-536

[4]. SLF4J, http://www.slf4j.org

[5]. Apache Log4j 2, http://logging.apache.org/log4j/2.x/

[6]. Equinox Log Service (org.eclipse.equinox.log)

*Add references simply by adding new items. You can then cross-refer to them by chosing <Insert><Cross Reference><Numbered Item> and then selecting the paragraph.* ***STATIC REFERENCES (I.E. BODGED) ARE NOT ACCEPTABLE, SOMEONE WILL HAVE TO UPDATE THEM LATER, SO DO IT PROPERLY NOW.***

## 6.2 Author’s Address

| | |
|---|---|
| Name | Evgeni Grigorov |
| Company | ProSyst Software |
| Address | Aachenerstr. 222, 50935 Cologne, Germany |
| Voice | +49 221 6604 501 |
| e-mail | e.grigorov@prosyst.com |

## 6.3 End of Document