RFC 234 - Framework Logging

Final

12 Pages

Abstract

Currently there is no API to define how the framework should do logging; code launching a framework needs to know implementation details when it wants to configurate that launching. This RFC proposes a framework implementation independent way to be used by the framework for logging.

0   Document Information

0.1   License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the "Distribution") in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. "OSGi Name Space" shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

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 10.1.

```
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	March 15 2017	Initial version based on #2974 and F2F discussions in Montpellier Carsten Ziegeler, Adobe Systems Incorporated, **cziegele@adobe.com**
1	April 11 2017	Changed to debug logging only after CPEG call Carsten Ziegeler

1 Introduction

This RFC defines a way to configure the logging of the framework itself independent from the framework implementation.

2 Application Domain

The launch API allows to start a framework in an implementation independent way. While this works, there is at least one thing which needs to be setup depending on the implementation: logging of the framework. As framework logging is an essential part of an application, the current launch API does not provide a full abstraction.

3   Problem Description

The framework launch API defines an implementation independent way to launch any framework implementation. This API covers creating a framework and configuring it. However, one aspect is not covered by this AP: Logging of the framework. This requires launch code like Apache Karaf to use implementation dependent solutions.

For example, by default the Apache Felix framework is logging to system out. But an implementation can provide a org.apache.felix.framework.Logger instance either through the framework properties (tricking the compiler) or by using reflection and checking for a setLogger method on the framework object. Eclipse Equinox is using the embedded LogService to log errors and has a separate tracing mechanism which eventually logs to system out as well.

Usually code that launches a framework also wants to tell the framework in what detail it should log and how to log.

4   Requirements

L0010 - The solution MUST define a way to set the ~~log detail (log level)~~domains for debug logging used by the framework implementation when emitting debug log messages.

L0020 – The solution MUST provide a way for the framework to detect the current ~~log level~~enabled domains for debug logging to avoid unnecessary calls into the logging mechanism.

L0030 – The solution MUST provide a way to capture the debug log messages emitted by the framework.

5   Technical Solution

The technical solution describes two parts, a logger instance created by the code launching the framework and how to pass that instance to the framework.

5.1   The **Debug**Logger

An instance of the ~~Debug~~Logger is created by the launching code. Initially a logger is created with a ~~log leve~~set of enabled domains for debug information~~l~~. ~~The log level can be changed by calling the~~ ~~setLogLevel~~ ~~method.~~ Before the framework logs, it ~~can~~should use the get~~Domains~~LogLevel method or ~~one of~~ the is~~XXX~~Enabled method~~s~~ to detect whether a log message~~s~~ should be emitted. The implementation of the DebugLogger.log method must check if the domain is enabled and discard the log message if the domain is not enabled. The framework can use ~~one of the two~~the provided log method~~s~~ to emit a log message~~.~~: The log method takes the domain as a first argument. The second argument is a string which supports parameterized logging. The token {} can be used to insert the value of the corresponding parameter passed through the args array instead of the token. If the last object in the args array is an exception/throwable this will be logged as an exception in addition to the message.

```
public ~~abstract class~~ interface ~~Debug~~Logger {

    public enum ~~LogLevel~~Domain {

        CLASSLOADER~~ERROR~~,

        ~~WARN~~MODULES,

        ~~INFO~~SERVICES,

        ~~DEBUG~~RESOLVER,

        SECURITY,

        STARTLEVELS

    }

    ~~public Logger(LogLevel l) {...}~~

    ~~public void setLogLevel(LogLevel l) { … }~~

    public ~~LogLevel~~EnumSet<Domain> get~~LogLevel~~Domains()~~;~~ ~~{ … }~~

    public boolean is~~Debug~~Enabled(Domain d)~~;~~ ~~{ … }~~

    ~~public boolean isInfoEnabled() { … }~~

    ~~public boolean isWarnEnabled() { … }~~

    public void log(~~LogLevel~~Domain ~~level~~d, String ~~msg~~fmt, Object... args)~~;~~ ~~{ … }~~

    ~~public final void log(LogLevel level, String msg, Throwable throwable) {...}~~

    ~~protected abstract void doLog(LogLevel level, String msg, Throwable throwable);~~

}
```

~~The Logger class is an abstract class and already implements all the functionality except for processing the log~~ ~~message from the framework. The~~ ~~log~~ ~~methods check the log level and call the abstract method~~ ~~doLog~~ ~~if the~~

level is enabled. Code launching a framework just needs to create an implementation subclass of DebugLogger and implement this single methodthe methods.

5.2 Providing the Logger to the Framework

The FrameworkFactory interface gets a new method to construct a framework which in addition to the configuration map gets a logger instance. The logger instance is optional. Calling the existing method newFramework(Map) is equivalent with calling this new method passing in null as the second argument:

```
public interface FrameworkFactory {

    Framework newFramework(Map<String, String> configuration, DebugLogger logger);

}
```

Whenever the framework wants to emit a debug log message it should check the current log level of the logger instance and only emit a message if the leveldomain is enabled.

If no logger instance is provided to the framework it is up to the framework implementation how to deal with debug log messages from the framework implementation.

5.3 Required Framework Logging

If a debug logger instance is provided to the framework factory, the framework implementation must use this instance to log the following events:emit debug log messages in the enabled domains. It is up to the framework implementation to decide when to emit a debug log message though. An implementation might decide to not debug log at all.

Whenever an ERROR framework event occurs, this must be logged with log level ERROR.

6 Data Transfer Objects

This RFC does not define any DTOs

7   Java API

8 Considered Alternatives

9   Security Considerations

This RFC does not need any new security considerations.

10 Document Support

10.1 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]   Data Transfer Objects, Core Release 6

10.2 Author's Address

Name	Carsten Ziegeler
Company	Adobe Systems Incorporated
Address	Barfüsserplatz 6, 4055 Basel, Switzerland
Voice	+41 61 226 55 0
e-mail	cziegele@adobe.com

10.3 Acronyms and Abbreviations

10.4 End of Document