RFP 159 JavaScript Microservices

Draft

8 Pages

Abstract

Discusses the need for a specification to make JavaScript a first-class target for OSGi Microservice concepts.

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 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	04/08/13	Simon Kaegi, Initial Draft

1 Introduction

This RFP discusses the need and requirements for a JavaScript implementation of the OSGi service layer. JavaScript is increasingly being used as a systems language and the lack of an inter-operable and well delineated story for service find, bind, and use is leading to wasted effort and re-invention of concepts already well understood in the OSGi community.

Ideas around "Universal OSGi" have been discussed since R3 and earlier. The current JavaScript concepts being used for this RFP come from the JavaScript OSGi Microservice implementation used by the Eclipse Orion project. This implementation was in turn based on earlier work done as a part of Eclipse's e4 cross-language support.

2 Application Domain

From its humble origins in 1996 JavaScript has ridden the success of the web browser to become one of the most ubiquitous and commonly used programming languages. In particular, the last five years have seen huge improvements in language performance and more recently JavaScript has become an increasingly popular choice as a server-side and general purpose systems language.

With this recent upbeat in interest the JavaScript community has been very actively developing new libraries and frameworks. Concerns like modularity are now important and there is broad consensus around the use of CommonJS “require” as its basis with two main variants: Node.js require and AMD (Asynchronous Module Definition). “require” is a building block for modularity and provides functionality similar to Java class-loading or C style imports. There is an excellent possibility JavaScript will provide first-class language level support for modules in EcmaScript 6, the next release of the language.

“require” places no strict requirement on packaging however when creating a library one frequently “packages” several modules together. This concept of packages has in particular taken hold in the Node.js community where NPM is used as the “node package manager”. Node packages are named and versioned and can declare dependency on another named and versioned package however unlike OSGi bundles do not provide API metadata. Similar to Java's JAR hell and Windows DLL Hell there is the concept of NPM hell to describe problems created with versioning inter-operabilities.

JavaScript is not multi-threaded and also does not provide co-routine support. It's processing model is event-loop based where each turn has run to completion semantics. As a result any blocking operations (for example synchronous use of XMLHttpRequest) are considered an anti-pattern and dangerous at best. Subsequently non-blocking operations with callbacks called upon completion are the norm.

The use of callbacks in a chain of requests for continuation passing makes code complex and hard to debug and understand. One technique to simplify callback use that is getting widespread recognition is the use of “Promises”, an object that represents an operations eventual result. There is effort to standardize cross-library Promises behavior called “Promises A+” and progress being made to support this interface in browsers in an effort called DOMFutures.

For the area of extensibility there have been a few efforts to standardize a service interaction model however they have been unsuccessful in reaching broad consensus. Most large JavaScripts projects that rely on having extensibility end up rolling their own variation on the service register, find, and bind model. This has limited inter-operability with other service models and resulted in a cottage industry with (in some cases) countless implementations of the same basic functionality. Tool vendors feel this acutely as they have to adapt to each frameworks ideas around plug-ability and extensibility.

2.1 Terminology + Abbreviations

JavaScript Microservices – name given to describe a JavaScript implementation of the OSGi Service Layer

Promise – a technique used to manage the complexity of non-blocking operation callbacks. An object that represents an operations eventual result and allows register callbacks and errbacks.

Require – A general term to describe the current state of the art technique for JavaScript module loading.

3 Problem Description

There is a lack of standardization around providing meaningful extensibility in JavaScript programs. Improved modularity like what is offered by “Require” is part of the solution but by itself insufficient as it implies coupling to a named module. Ideally there should be no direct coupling to the module providing the implementation and instead just loose coupling to a provider API or service name.

Getting the details around creating an extensibility API correct is difficult and efforts at OSGi show it requires iteration, and trial and error to get right. Even once one gets the details of such an extensibility API correct there is the challenge to ensure that it is simple to use, easy to understand and implement against.

4 Use Cases

4.1 Web-based Tooling Platform

Developer tooling platforms require strong support for extensibility in order to allow handling all manner of differences in language tooling and parsing, source control, UI customization, key-bindings, editor settings and themes to name just a few. The Eclipse Orion project is committed to using services and service references as the primary means of extensibility. The hope is that by using standards-based interfaces other tool vendors could more efficiently contribute to the developer ecosystem without having to re-adapt code to their own unique system.

4.2 Node Application Services

Its currently a challenge to write back-end JavaScript systems that can support variability on service implementation. In nearly all cases there are significant API difference between two libraries offering similar functionality. It would be tremendously helpful to facilitate specification of back-end service APIs and provide a consumption model.

4.3 Service implementations in alternate languages

Many companies have very significant investment in Java that they would like to continue to leverage. By decoupling the service implementation from the requirement we allow the possibility to re-use existing Java-technology without having to rewrite everything. Two examples where this might be useful are: Node.js based systems where Java might be used to provide some service implementation, Eclipse IDE where we might want to use language tooling written in JavaScript .

5 Requirements

5.1 Basic

- BA001 - Create a specification that provides a JavaScript binding of the OSGi Service Layer capabilities
- BA002 – Must provide API for registering a service with a set of attribute values that describe the service
- BA003 – Must provide API for looking up a service. Service lookup must have the ability to filter on the attributes of a service
- BA004 – Must provide API for listening to service events. Service listening must have the ability to filter on attributes of the service associated with the service event
- BA005 – The API should not preclude creation of very small implementations
- BA006 – The API should hide the synchronous/asynchronous nature of a service implementation.
- BA007 – The specification must not require a specific framework or host environment other than an EcmaScript 5 compliant JavaScript run-time.

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

6.2 Author's Address

Name	Simon Kaegi
Company	IBM
Address	770 Palladium Drive, Kanata, Ontario, Canada
Voice	(613) 270-4821
e-mail	Simon_Kaegi@ca.ibm.com

6.3 End of Document