RFP 180 - IoT Protocols: MQTT

Final

8 Pages

Abstract

In the IoT domain there is a widespread of communication protocols available for letting devices interact with eachother. A popular publish-subscribe protocol for communicating with IoT devices is MQTT. This RFP focuses on ways to integrate the MQTT protocol with OSGi. The goal to bridge MQTT to the existing OSGi EventAdmin.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

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	*January 7 2016*	*Initial contribution – extracted from initial RFP 177* *Tim Verbelen, iMinds – Ghent University, tim.verbelen@intec.ugent.be*
0.1	*January 14 2016*	*Updates F2F Madrid* *Tim Verbelen, iMinds – Ghent University, tim.verbelen@intec.ugent.be*
Final	*May 2016*	*Tim Verbelen, mark as final*

1 Introduction

Internet of Things (IoT) is becoming an important application domain of OSGi. The ability to run an OSGi framework on a gateway device as well as a Cloud server, together with the ability of transparently calling remote services using distributed OSGi makes it perfect base for an IoT platform. However, the proliferation of IoT protocols makes it difficult to integrate the many technologies available. One popular IoT protocol for connecting such IoT devices is MQTT. This RFP provides a solution to transparently handle MQTT events in OSGi via the EventAdmin.

2 Application Domain

In the current IoT domain we see a proliferation of different protocols for device access, remote management and IoT applications.

Various device access protocols are defined to interface with small (wireless) sensor devices, such as enOcean, ZigBee, Z-Wave, etc. Also for device management different protocols exist such as OMA DM, TR-69, etc. While these protocols are already being handled in the OSGi specification (103 – Device Access, 117 – Dmt Admin, 141- Device Abstraction layer, and specific protocol adapters for enOcean, TR069, …), there are also many IoT protocols that are widely used on an application level (i.e. CoAP, MQTT).

One of the more popular application-level IoT protocols nowadays is the event based MQTT.

2.1 MQTT

MQTT is a lightweight publish/subscribe messaging protocol on top of TCP/IP standardized by OASIS [3]. MQTT uses a central broker where all clients send their messages to and where clients can subscribe on certain topics.

MQTT topics are represented as hierarchically structured strings with a forward slash topic separator, i.e. “sensors/temperature/mysensor1”. To subscribe to topics, filters can be used containing wildcard characters. The hash sign '#' is used as a multi-level topic wildcard, the plus sign '+' is used as a single level wildcard. Topic names starting with a dollar sign '$' are special topics that relate to the event broker, and clients should not use topic names starting with '$' to exchange messages with other clients.

The MQTT broker supports multiple QoS levels: QoS level 0 : best effort delivery: the message is not acknowledged by the receiver or stored and redelivered by the sender. QoS level 1: the message is guaranteed to be delivered at least once to the receiver. The sender stores the messages until it gets an acknowledgement. QoS level 2: the message is delivered exactly once to the receiver. This is the highest and slowest QoS level.

MQTT also supports so called “last will and testament” messages, which are sent out by the broker when the sender loses connectivity with the broker.

The MQTT specification does not specify any format the message payload should adhere to. However, the application developer will know the payload format, as this is implicitly defined by the topic(s) he subscribes to.

3 Problem Description

Although there are already solutions for using MQTT within an OSGi application, these still leave it up to the developer to use the specific APIs of the MQTT library used, which is currently not standardized in OSGi. For

example the Eclipse Paho [4] MQTT client API is used in the Eclipse Kura [5] platform. To facilitate interaction with an MQTT broker, this RFP aims to bridge MQTT and EventAdmin:

- subscribe to an MQTT broker topic by registering an EventHandler in the OSGi framework

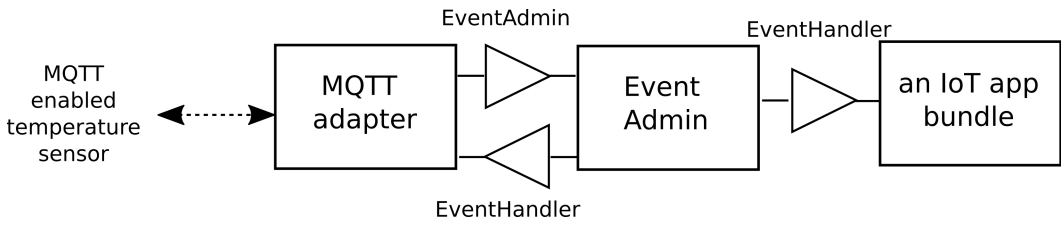
- sending out MQTT messages by publishing an Event via EventAdmin

4 Use Cases

4.1 Integrate with small, embedded devices unable to run OSGi

Many devices are powered with a small microcontroller (e.g. Arduino-like) that is unable to run a full OSGi stack, but often does support communicating over IoT protocols such as MQTT.

For example a small temperature sensor connects to an MQTT broker and sends out MQTT events with the temperature value on topic “sensor/temperature”. Withing OSGi, an IoT application bundle wants to listen to these events. This bundle registers an EventHandler with topic “sensor/temperature”. An MQTT adapter bundle is configured with a MQTT broker URI and a set of topics to bridge to MQTT. Once the EventHandler is registered, the MQTT adapter will subscribe to this topic with the MQTT broker, and post any received MQTT message to the EventAdmin.



Similarly, one could also forward Events posted on the EventAdmin to the MQTT broker.

Since MQTT does not provide any structure for the payload, it is up to the application developer to correctly parse the MQTT payload.

4.2 MQTT as distributed eventing mechanism between OSGi frameworks

An adapter that translates MQTT events to OSGi events through EventAdmin and vice versa could provide a mechanism for distributed eventing in OSGi.

In this case, the MQTT message payload could be the JSON String representing the OSGi event using the Object Conversion spec.

5 Requirements

5.1 MQTT

- M0010 – The solution MUST define a mapping from an MQTT topic to an EventAdmin topic
- M0020 – The solution MUST define a mapping from an MQTT message to an OSGi Event
- M0030 – The solution SHOULD provide a mechanism to define which OSGi Events have to be sent to the MQTT broker and which EventAdmin topics have to be subscribed with the MQTT broker
- M0040 – The solution MUST be configurable (i.e. which MQTT broker to connect to)
- M0050 – The solution MUST support the different MQTT QoS levels
- M0060 – The solution MUST support providing last will and testament messages

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]. MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. 29 September 2014. OASIS Standard. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[4]. Eclipse Paho, https://www.eclipse.org/paho/

[5]. Eclipse Kura, https://www.eclipse.org/kura/

6.2 Author's Address

Name	Tim Verbelen
Company	iMinds, Ghent University
Address	Gaston Crommenlaan 8 box 201, 9050 Ghent, Belgium
Voice	
e-mail	tim.verbelen@intec.ugent.be

6.3 End of Document