RFP 177 - IoT Protocols: CoAP

Final

9 Pages

Abstract

In the IoT domain there is a widespread of communication protocols available for letting devices interact with eachother. One popular protocol for communicating with devices on an application level is CoAP (Constrainted Application Protocol). This RFP focuses on ways to integrate the CoAP protocol with OSGi. The goal to allow low-end devices that are unable to run a complete OSGi stack communicate with an OSGi framework using CoAP.

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	*October 9 2015*	*Initial contribution* *Tim Verbelen, iMinds – Ghent University, tim.verbelen@intec.ugent.be*
0.1	*January 7 2016*	*Focus on the CoAP protocol only, split off MQTT* *Tim Verbelen, iMinds – Ghent University, tim.verbelen@intec.ugent.be*
0.2	*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, there are still a class of embedded devices that are unable to run an OSGi runtime but that have to be able to communicate with an OSGi framework. One popular IoT protocol for connecting such small device is CoAP. This RFP provides a solution to set up communication between an OSGi framework and devices talking CoAP.

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. MQTT, CoAP). This RFP targets the latter.

2.1 CoAP

One of the more popular application-level IoT protocols nowadays is CoAP [3]. The Constrained Application Protocol (CoAP) is standardized by the IETF and is intended to provide a lightweight protocol for machine to machine communication. It is a RESTful protocol that was designed to provide transparent mapping to HTTP. Devices (also called “Resources” in CoAP) are made available through URIs, and clients access these using GET, PUT, POST and DELETE methods. It uses UDP as transport layer in order to limit bandwidth and overhead.

Multiple CoAP resources can be made available on a single endpoint. The available resources on a given endpoint are made available on the /.well-known/ path prefix using the CoRE Link Format. Different CoAP endpoints can also be discovered by using the following “All CoAP Nodes” multicast addresses: 224.0.1.187 for IPv4 networks and FF0x::FD for IPv6 networks. In case of IPv6 CoAP only listens to the Link-Local and Site-Local scopes.

CoAP supports a limited number of content formats and encodings to be used for the payload. To minimize the overhead the content format is set in the header using a numeric identifier. The supported content formats are:

- text/plain;charset=utf-8, ID 0
- application/link-format, ID 40
- application/xml, ID 41
- application/octet-stream, ID 42
- application/exi, ID 47
- application/json ID 50

The client can also provide the Accept option in his request, indicating which content format is acceptable for the client. If no Accept option is given, the client does not express a preference (no default). If the client does provide an Accept option, the server returns the preferred content format, or a 4.06 “Not Acceptable” error is sent.

Standard CoAP hence only supports a pull-based model, where the client has to perform a GET request to get the status of a certain device. This does not work well when a client wants to receive updates when the device state changes. To cope with this, IETF has proposed the Observe extension to CoAP [4], where the client can set an Observe option. Then the server will keep on sending notifications each time the resource changes.

2.2 Device semantics on top of CoAP

CoAP however does not standardize any semantics, such as how physical devices should map on CoAP resources, which URIs have to be provided, or what the data represents in the payload. Other initiatives exist to tackle this, such as IPSO Application Framework [5] and OMA Lightweight M2M [6].

The IPSO Application Framework defines a set of RESTful profiles modeling IP smart objects. The framework may be used over either HTTP or CoAP. An IPSO profile consists of a group of resource types, called function sets, that define REST interfaces that represent an object. There are currently function sets available for Device, GPIO, Power, Sensors, Light Control, Message and Location. For example the Light function set is defined as:

```
+---------------+-------------+-------------+----+------+---------+
| Type          | Path        | RT          | IF | Type | Unit    |
+---------------+-------------+-------------+----+------+---------+
| Light Control | /lt/{#}/on  | ipso.lt.on  | a  | b    |         |
| Light Dimmer  | /lt/{#}/dim | ipso.lt.dim | a  | i    | 0-100 % |
+---------------+-------------+-------------+----+------+---------+
```

This specifies a path, a IETF specified CoRE resource type, a CoRE interface (a = actuator), a datatype (b = boolean, I = integer) and optionally a unit. Return values are represnted using SenML (Sensor Markup Language) which is serialized to JSON or XML.

OMA Lightweight M2M also defines an object and resource model for representing application semantics on top of CoAP. An object is a collection of resources. A resource is an atomic piece of information that can be read, written or executed. Objects/Resources can be accessed with URIs of the format: /{Object ID}/{Object instance}/{Resource ID}. Object ids and resource ids are integers that are registered with the OMA naming authority, specifying all kinds of actual object types. Payloads are plain text for individual resources, or JSON for resource batches.

For example the LWM2M Location object (Object ID 6) consists of 6 resources:

Resource name	ID	Access Type	Multiple instances?	Type	Range	Unit	Description
Latitude	0	R	No	Decimal		Deg	...
Longitude	1	R	No	Decimal		Deg	...
Altitude	2	R	No	Decimal		m	...
Uncertainty	3	R	No	Decimal		M	...
Velocity	4	R	No	Decimal		m/s	...
Timestamps	5	R	No	Time			...

Hence URI 6/1/0 returns the latitude of Location object with instance 1

2.3 Device Access specification

OSGi already has a Device Access specification specification that supports the coordination of automatic detection and attachment of existing devices on an OSGi Framework. This could also be leveraged to represent CoAP enabled devices in OSGi by specifying a CoAP device category with an interface consisting of get, put, post and delete methods.

In case a semantic layer such as LWM2M is used, a refining driver could map the CoAP device to a more higher level device category.

2.4 Device Abstraction Layer specification

The Device Abstraction Layer specification provides a unified interface for application developers to interact with sensor, devices, etc. connected to a gateway. Application developers don't have to deal with protocol specific details which simplifies the development of their applications.

Again, in case semantic information about the CoAP device is available, this could be mapped onto the DAL Device and Function interfaces.

3 Problem Description

This RFP seeks to provide a way to let an OSGi framework communicate with small IoT devices that are unable to run an OSGi runtime but are connected to the network via CoAP. On the one hand, these devices can be made available as a service in an OSGi runtime. On the other hand, an OSGi service could make services available as CoAP resources that can be called from such embedded devices. Therefore, this RFP makes it possible to:

- expose an OSGi service as a CoAP resource to the outside world

- represent a remote CoAP resource as a local service within the OSGi framework

4 Use Cases

4.1 Import CoAP device as OSGi service

A CoAP device is known to be on a certain URI. To make this device available inside the OSGi framework one registers a Device service with device category CoAP and the URI as service property. The Device Access bundle will search for a CoAP driver, that then registers a CoAPDevice service. When the CoAP device follows a semantic model such as LWM2M, a refining driver can provide more high level interfaces representing the device, for example using the Device Abstraction Layer interfaces.

4.2 Thermostat microcontroller calls weather service

Suppose a thermostat microcontroller that regulates the temperature inside a house. The microcontroller is unable to run an OSGi stack, but is connected to the Internet using a lightweight CoAP stack. To better control the temperature, the termostat wants to call a weather service that predicts the outside temperature. This weather service is implemented as an OSGi service. A CoAP provider bundle can map the service name and methods to URIs, so that the OSGi service is made available as a CoAP resource, and can be called by the thermostat. The way of mapping methods to URIs could be similar as described for REST and JSONRPC in RFP 172.

5 Requirements

5.1 Import CoAP devices

- I0010 – The solution MUST define a CoAP device category
- I0020 – The solution MUST define a CoAPDevice interface and mandatory/optional service properties.
- I0030 – The solution MAY define refining device categories on a semantically higher level
- I0040 – The solution MAY provide a mechanism to discover CoAP endpoints and their resources
- I0050 – The solution MAY support the CoAP Observe extension
- I0060 – The solution MAY provide a CoAP request builder API

5.2 Export CoAP resources

- E0010 – The solution MUST define a mapping from a Java interface to a CoAP resource
- E0020 – The solution MUST define on what endpoint the URI will rest.
- E0030 – The solution MAY allow the service to optionally define the actual endpoint URI.
- E0040 – The solution MUST define how the URI segments are mapped and converted to the method's parameters.
- E0050 – The solution MUST define how the returned object is converted to a payload format for the client, taking into account the Accept option if set by the client.

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]. IETF RFC 7252: The Constrained Application Protocol (CoAP), Z. Shelby, K. Karte, C. Bormann, https://tools.ietf.org/html/rfc7252

[4]. IETF RFC 7641: Observing Resources in the Constrained Application Protocol (CoAP), K. Karte, https://tools.ietf.org/html/rfc7641

[5]. The IPSO Application Framework, Z. Shelby, C. Chauvenet, http://www.ipso-alliance.org/wp-content/uploads/2016/01/draft-ipso-app-framework-04.pdf

[6]. OMA Lightweight M2M v1.0, http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m-v1-0

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