

Service Layer API for oneM2M

Draft

51 Pages

Text in Red is here to help you. Delete it when you have followed the instructions.

The <RFC 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 automatically when the document is saved.

Abstract

10 point Arial Centered.

oneM2M is standard organization and specifies middleware for IoT, called Common Services Entities (CSE). Application can access functionality in CSE with RESTful operations, which are Create, Retrieve, Update, Delete and Notify. oneM2M allows variety of communication methods, 4 protocol bindings (HTTP, MQTT, CoAP, Websocket) and 3 serializations (XML, JSON, CBOR). This RFC describes the way to provide high level API for oneM2M RESTful operations hiding the difference of variety of communication methods.



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,



2017年9月14日

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

0.1 License 2 0.2 Trademarks 3 0.3 Feedback 3 0.4 Table of Contents 3 0.5 Terminology and Document Conventions 4 0.6 Revision History 4 1 Introduction 5 2 Application Domain 5 2.1 IoT Application configuration using oneM2M 5 2.2 Communication methods used in oneM2M 6 2.3 Long name and short name 7 3 Problem Description 7 4 Requirements 7 5 Technical Solution 8 5.1 Overview for communication through network 8 5.2 Overview for internal communication within single OSGi framework 10 5.3 Service Property for sub-interfaces of Service Layer Interface 10 5.4 ClientLibrary 11 5.5 Validator Interface 13	0 Document Information	2
0.2 Trademarks 3 0.3 Feedback 3 0.4 Table of Contents 3 0.5 Terminology and Document Conventions 4 0.6 Revision History 4 1 Introduction 5 2 Application Domain 5 2.1 IoT Application configuration using oneM2M 5 2.2 Communication methods used in oneM2M 6 2.3 Long name and short name 7 3 Problem Description 7 4 Requirements 7 5 Technical Solution 8 5.1 Overview for communication through network 8 5.2 Overview for internal communication within single OSGi framework 10 5.3 Service Property for sub-interfaces of Service Layer Interface 10 5.4 ClientLibrary 11		
0.3 Feedback 3 0.4 Table of Contents 3 0.5 Terminology and Document Conventions 4 0.6 Revision History 4 1 Introduction 5 2 Application Domain 5 2.1 IoT Application configuration using oneM2M 5 2.2 Communication methods used in oneM2M 6 2.3 Long name and short name 7 3 Problem Description 7 4 Requirements 7 5.1 Overview for communication through network 8 5.2 Overview for internal communication within single OSGi framework 10 5.3 Service Property for sub-interfaces of Service Layer Interface 10 5.4 ClientLibrary 11		
0.4 Table of Contents 3 0.5 Terminology and Document Conventions 4 0.6 Revision History 4 1 Introduction 5 2 Application Domain 5 2.1 IoT Application configuration using oneM2M 5 2.2 Communication methods used in oneM2M 6 2.3 Long name and short name 7 3 Problem Description 7 4 Requirements 7 5.1 Overview for communication through network 8 5.2 Overview for internal communication within single OSGi framework 10 5.3 Service Property for sub-interfaces of Service Layer Interface 10 5.4 ClientLibrary 11		
0.5 Terminology and Document Conventions	0.4 Table of Contents	3
0.6 Revision History		
2 Application Domain		
2 Application Domain	1 Introduction	5
2.1 IoT Application configuration using oneM2M		
2.2 Communication methods used in oneM2M 6 2.3 Long name and short name 7 3 Problem Description 7 4 Requirements 7 5 Technical Solution 8 5.1 Overview for communication through network 8 5.2 Overview for internal communication within single OSGi framework 10 5.3 Service Property for sub-interfaces of Service Layer Interface 10 5.4 ClientLibrary 11	2 Application Domain	5
2.3 Long name and short name	2.1 lo l Application configuration using oneM2M	5
3 Problem Description		
4 Requirements	2.3 Long name and short name	/
5 Technical Solution	3 Problem Description	7
5.1 Overview for communication through network	4 Requirements	7
5.1 Overview for communication through network	5 Tochnical Solution	8
5.2 Overview for internal communication within single OSGi framework		
5.3 Service Property for sub-interfaces of Service Layer Interface		
5.4 ClientLibrary11		



illiance	Draft	2017年9月1	L4 日
6 Data Transfer Objects		13	
6.2 RequestDTO		14	
6.6 AttributeDTO		15	
7 Javadoc		16	
8 Considered Alternatives		49	
8.1 Representation of DT0	O	49	
	or receiving notification by AE		
8.3 Non blocking API for (ClientLibrary	50	
9 Security Considerations		50	
9.1 ProtocolBinding Servi	ce with secure protocols	50	
9.2 Using Multiple Certific	ates with in a single ProtocolBinidng Service	50	
10 Document Support		50	
	eviations		
10.4 End of Document		51	

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	SEP 19 2017	Put information relating to the changes you have made here. <name>, <company> <e-mail> Initial Contribution. Hiroyuki Maeomichi, NTT, maeomichi.hiroyuki@lab.ntt.co.jp</e-mail></company></name>

Introduction

Introduce the RFC. Discuss the origins and status of the RFC and list any open items to do.

oneM2M is standard organization and specifies middleware for Internet of Things (IoT), called Common Services Entities (CSE). Applications can access CSE's functionality with RESTful operations, which are Create, Retrieve, Update, Delete and Notify. TS-0001 [2] defines more than 40 resource types to expose CSE's functionalities. oneM2M allows variety of communication methods, combination of 4 protocol bindings (HTTP, MQTT, CoAP, Websocket) and 3 serializations (XML, JSON, CBOR).

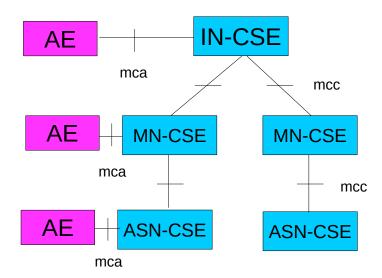
This RFP discuss the way to provide high level API (namely service layer API) for oneM2M RESTful operations hiding the difference of variety of communication methods.

2 Application Domain

This section should be copied from the appropriate RFP(s). It is repeated here so it can be extended while the RFC authors learn more subtle details.

2.1 IoT Application configuration using oneM2M

oneM2M's middleware, called CSE can be deployed in different locations and they are connected each other forming tree topology. Depending on deployed location, CSEs are categorized to 3 types, IN-CSE, MN-CSE and ASN-CSE. IN-CSE is located top of tree, ASN-CSE is located at leaf and MN-CSE is located and MN-CSE is located on middle.



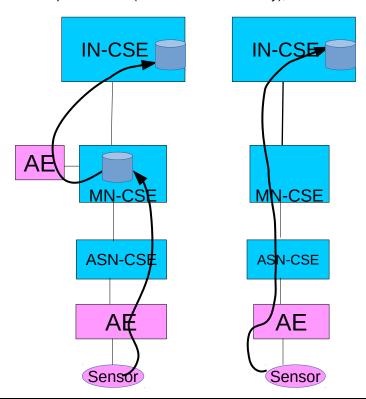


oneM2M's application, called Application Entity (AE) connects to one of CSEs. After AE connecting to the CSE, AE can access to all of CSEs, by retargeting function of CSE.

AE accesses to CSE's functionality through RESTful API, which consists of Create, Retrieve, Update, Delete and Notify in targeting more than 40 types of resources. For examples, typical resources are < contentInstance> that expresses IoT data and <container> that holds set of <contentInstance>s. AE can create or retrieve the <contentInstance> on any CSE by the retargeting functionality, as far as permission is allowed. Interface between CSEs is called mcc and interface between CSE and AE is called mca, both interfaces have almost same interface.

It is possible to develop variety types of distributed applications using the architecture. For example for IoT data aggregation applications, it is possible to develop gradual aggregation type or direct aggregation type. In gradual aggregation type, AE connected to ASN-CSE creates <conentInstance>s in ASN-CSE, and intermediate applications calculate statistics and put the result on IN-CSE as a <contentInstance>, while, in direct aggregation type, AE connected to ASN-CSE creates <contentInstance>s in IN-CSE directly.

Under CSE layer, oneM2M specifies NSE(Network Services Entity), but this RFC doesn't cover the NSE layer.



2.2 Communication methods used in oneM2M

oneM2M allows variety of communication methods, combination of 4 protocol bindings (HTTP, MQTT, CoAP, Websocket) and 3 serializations (XML, JSON, CBOR). It might be added in future. oneM2M specifies specification in different level.

Firstly TS-0001[2] specifies high level resource definitions, it defines more than 40 resource types, such as <contentInstance> for storing IoT data, <timeSeriesInstance> for periodic sensor measurement with leap detection mechanism.



2017年9月14日

Secondly TS-0004[3] specifies procedures and serializations in independent manner from protocol bindings. Resource type and protocol data unit are defined using XSD for XML serialization. Mapping between XML and other serializations are also specified.

Thirdly TS-0008, TS-0009, TS-0010, TS-0020 specify protocol specific details for CoAP, HTTP, MQTT and Web Socket respectively.

2.3 Long name and short name

oneM2M introduced two types of notation, called long name and short name for resource types, attribute and so on. Long name is human friendly string and specifications mainly use this notation, while short name is short string consist of typically 2 or 3 characters (but not limited and sometimes longer) and communication protocol use this notation. In most cases, the initial characters of long name are assigned as short name, for examples, ct for CreationTime and at for AnnounceTo.

3 Problem Description

This section should be copied from the appropriate RFP(s). It is repeated here so it can be extended while the RFC authors learn more subtle details.

oneM2M specifies protocol based interface, but doesn't specify programing level API. As previously mentioned oneM2M allows variety of communication methods which are the combinations of 4 protocol bindings (HTTP, MQTT, CoAP, Websocket) and 3 serializations (XML, JSON, CBOR).

First problem is application portability. Without standardized API, application program tends to depend on the communication method initially intend to use and it will became hard to run another environment in which uses another communication method. (For example, an application designed for XML/HTTP, tend to run on environment use JSON/Websocket)

Second problem is the latency of the communication between CSE and application. Even if CSE and application is located in the same box, current oneM2M specifications define methods through protocols which requires serialization/deserialization of data, context-switch of applications, validation of incoming data and resulted in large latency compared to the situation both CSE and Application resides in the same Java VM and communicate with Java interfaces. Large latency reduces applicable area of oneM2M based solution.

Third problem is the complexity of handling of long name and short name. Even if short name is defined by trying to use initial characters, it is not straight forward to translate them in head.

4 Requirements

This section should be copied from the appropriate RFP(s)



2017年9月14日

- R0010 The solution MUST provide means to access outer CSE from application.
- R0011 The solution MUST provide means to access outer CSE from client CSE.
- R0012 The solution MUST provide means to select a communication method for application.
- R0013 The solution MUST provide means to select a communication method for client CSE.
- R0020 The solution MUST provide means for CSE to accept requests form outer CSE.
- R0020 The solution MUST provide means for CSE to accept requests form outer application.
- R0030 The solution MUST provide means to communicate through Java interface between CSE and application that are located in the same OSGi framework.
- R0040 The solution SHOULD hide differences of communication methods, which are combinations of 4 protocol bindings and 3 serializations (XML, JSON, CBOR).
- R0050 The solution SHOULD provide developer friendly way for handling short names.
- R0060 The solution MUST provide asynchronous interface using 'call by value', such as DTO.

5 Technical Solution

First give an architectural overview of the solution so the reader is gently introduced in the solution (Javadoc is not considered gently). What are the different modules? How do the modules relate? How do they interact? Where do they come from? This section should contain a class diagram. Then describe the different modules in detail. This should contain descriptions, Java code, UML class diagrams, state diagrams and interaction diagrams. This section should be sufficient to implement the solution assuming a skilled person.

Strictly use the terminology a defined in the Problem Context.

On each level, list the limitations of the solutions and any rationales for design decisions. Almost every decision is a trade off so explain what those trade offs are and why a specific trade off is made.

Address what security mechanisms are implemented and how they should be used.

5.1 Overview for communication through network

Protocol binding service and Mapper service are introduced to handle different protocols and serializations, respectively. CSE communicates the protocol binding service through Service Layer Interface. The interface is protocol and serialization agnostic interface. Protocol binding service uses Mapper service to handle different serializations. For the Application entity, higher level abstraction is provided by Client Library Service through ClientLibrary interface. Since this service is a stateful service, so it will be generated by Client Library Factory. Following figure illustrates overall architecture.



AE ClientLibrary Client Client **CSE** Library Library Service **Factory** Service Layer create **Protocol Protocol** Mapper Binding Binding Service Service Service Mapper Network

Service Layer Interface is defined as follows. Only method request sends request message and return Promise for the response. Here, Promise enables asynchronous messaging.

On the Service Layer interface, there are bidirectional invocations, that is, CSE sends request to protocol binding service and get response from it, protocol binding service sends request to CSE and get response from it. For the



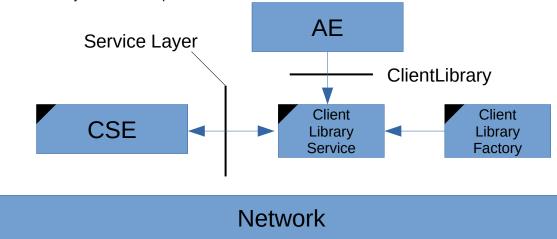
2017年9月14日

clarification purpose, Service Layer Interface is extended to ProtocolBinding Interface, Cse Interface, and Ae Interface without additional methods.

```
public interface ProtocolBinding extends ServiceLayer {}
public interface Cse extends ServiceLayer {}
public interface Ae extends ServiceLayer {}
```

5.2 Overview for internal communication within single OSGi framework.

For internal communication, Cse Service and ClientLibrary Service communicate directly without inter-mediating ProtocolBinding Service. Following figure depicts overall architecture. Though this type of communication is not clearly defined in oneM2M specification, communicating directly without serializing data between AE and CSE allows shorter latency and less computational resources.



5.3 Service Property for sub-interfaces of Service Layer Interface

Depending on the sub-interfaces of Service Layer Interface, they put different service properties. Table below summarizes the properties on the interfaces.

Interface	property Name	type	explanation
ProtocolBinidng	protocol	String	Supporting protocol. Possible values are "HTTP", "MQTT", "CoAP", or "WebSocket"
	serialization	String	Serialization. Possible values are "XML", "JSON" or "CBOR"
	Secure	boolean	True, if secure protocol is supported, otherwise false.
	versions	String[]	Supported versions, possible values would be "R1", "R1.1", "R2", "R2A".
Cse	CSE-ID	String	CSE-ID: ID of CSE
	SP-ID	String	ID of Service Provider
	CSE-type	String	Type of CSE. Possible values are IN, MN, or ASN
	POA	String	URI for point of access
	versions	String[]	Supported versions



2017	7年9	日	14	\Box

Ae	AE-ID-Stem	String	Head part of AE-ID
	APP-ID	String	Application ID
	POA	String	URI for point of access
	versions	String[]	Supported versions

5.4 ClientLibrary

For the Application entity, higher level abstraction is provided by Client Library Service through ClientLibrary interface. ClientLibrary manages Request-ID and AE-ID, so that it need to be stateful service. It will be generated by Client Library Factory depending on using AE bundles. ClientLibrary Interface provides more developer friendly interface that exposes CRUD+N operation on resources.

```
package org.osgi.onem2m.client;
public interface ClientLibrary {
       * Create resource
       * @param uri URI for the target Resource
       * @param resource ResourceDTO for creating resource
       * @return ResouceDTO for created resource
      public ResourceDTO create(String uri, ResourceDTO resource) throws OneM2MException;
       * Retrieve resource
       * @param uri URI for the target Resource
       * @return ResouceDTO for retrieved resource
      public ResourceDTO retrieve(String uri);
      /**
       * Retrieve of partial attributes.
       * @param uri URI for the target Resource
       * @param attributes attribute names for retrieving attribute.
       * @return ResouceDTO for retrieved resource
      public ResourceDTO retrieve(String uri, List<String> attributes) throws
OneM2MException;
       * Update resource
       * @param uri URI for resource
       * mparam resource resource data
       * @throws OneM2MException
```

2017年9月14日

Draft

```
*/
public ResourceDTO update(String uri, ResourceDTO resource) throws OneM2MException;

/**
    * Delete resource
    * @param uri URI for resource
    * @throws OneM2MException
    */
public void delete(String uri) throws OneM2MException;

/**
    * send notification
    *
    * @param poa
    * @param notification
    */
public void notify(NotificationDTO notification ) throws OneM2MException;
...
}
```

There are some configuration methods. Application must set communication method by calling setProtocol(), setSerialization(), setSecureProtocol() and some communication information by calling setAeId(), setRequestIDHeader(). If the Application need to receive notifications, it must set notification listener by calling setNotificationListener().

2017年9月14日

Draft

```
/**
 * set Application ID. This value is used for registration process of oneM2M.
 * "C" is used for the initial registration. After registration the assigned AEID.
 *
 * @param aeid
 */
public void setAeId(String aeid);

/**
 * set header for request ID
 *
 * @param header for request ID
 */
public void setRequestIDHeader(String format);

/**
 * set Notification Listener with client library.
 *
 * @param poa
 * @param listner
 */
public void setNotificationListener(String poa, NotificationListener listener );
...
}
```

NOTE: need to add setAppID()?

5.5 Validator Interface

Validator Services implementing Validator interface, can be used for validating OneM2MDTO and its sub classes.

TBD

6 Data Transfer Objects

RFC 185 defines Data Transfer Objects as a generic means for management solutions to interact with runtime entities in an OSGi Framework. DTOs provides a common, easily serializable representation of the technology.

For all new functionality added to the OSGi Framework the question should be asked: would this feature benefit from a DTO? The expectation is that in most cases it would.

The DTOs for the design in this RFC should be described here and if there are no DTOs being defined an explanation should be given explaining why this is not applicable in this case.



This section is optional and could also be provided in a separate RFC.

6.1 OneM2MDTO

OneM2MDTO is the base DTO for more concrete DTO related to oneM2M. The DTO holds a Map field and JSON like structure is stored in it.

6.2 RequestDTO

RequestDTO holds a Request Information used for oneM2M communication. This DTO extends OneM2MDTO but has no additional fields. The extension is used for clarification purpose of type.

```
package org.osgi.onem2m.dto;

/**
 * DTO containing oneM2M request primitive.
 */
public class RequestDTO extends OneM2MDTO {
}
```

6.3 ResponseDTO

ResponseDTO holds a Response Information used for oneM2M communication. This DTO extends OneM2MDTO but has no additional fields. The extension is used for clarification purpose of type.

```
package org.osgi.onem2m.dto;

/**
 * DTO containing oneM2M response primitive.
 */
public class ResponseDTO extends OneM2MDTO {
}
```



6.4 ResourceDTO

ResourceDTO holds information for oneM2M resource. Extended attribute called resourceType holds resource type in short name, like "AE".

6.5 NotificationDTO

NotificationDTO holds information for Notification. This DTO extends OneM2MDTO but has no additional fields. The extension is used for clarification purpose of type.

```
package org.osgi.onem2m.dto;

/**
 * DTO for Notification.
 */
public class NotificationDTO extends OneM2MDTO {
}
```

6.6 AttributeDTO

AttributeDTO holds information for attribute. Extended fields, type and attributeName has resource type and name of attribute respectively.



7 Javadoc

Please include Javadoc of any new APIs here, once the design has matured. Instructions on how to export Javadoc for inclusion in the RFC can be found here: https://www.osgi.org/members/RFC/Javadoc



Draft 2017年9月14日

Demo Documentation

17/09/14 19:26

Package Sum	mary	Page
org.osgi.onem2 m.client	Package containing interfaces of Client Library for oneM2M.	17
org.osgi.onem2 m.dto	Package containing Data Transfer Objects (DTOs).	26
org.osgi.onem2 m.mapper	Package contains Mapper interface that convert binary data to OneM2MDTO, vice versa.	34
org.osgi.onem2 m.servicelayer	The key Package containing Service Layer API.	36
org.osgi.onem2 m.util	Package containing Utility Classes for oneM2M.	41
org.osgi.onem2 m.validation	Package containing validation interface for oneM2M data structures.	45

Package org.osgi.onem2m.client

Package containing interfaces of Client Library for oneM2M.

See:

Description

Interface Sum	Interface Summary		
ClientLibrary		19	
CllientLibraryF actory		23	
NotificationList ener	Interface for Notification Listener White Board Pattern にする場合。	23	

Ex	tion Summary	Page
On tio	<u>ИЕхсер</u>	25

Package org.osgi.onem2m.client Description

Package containing interfaces of Client Library for oneM2M.

Interface ClientLibrary

org.osgi.onem2m.client

public interface ClientLibrary

Method	Summary	Pag e
ResourceDT O	<pre>create (String uri, ResourceDTO resource) Create resource</pre>	21
void	<pre>delete(String uri) Delete resource</pre>	22
String	<pre>getAeId() get AEID.</pre>	20
long	getRequestIDCount () get Counter for the RequestID.	21
void	<pre>notify (String poa, NotificationDTO notification) send notification</pre>	22
ResourceDT O	retrieve (String uri) Retrieve resource	21
ResourceDT O	<pre>retrieve (String uri, List<string> attributes) Retrieve of partial attributes.</string></pre>	21
void	<pre>setAeId (String aeid) set Application ID.</pre>	20
void	<pre>setNotificationListener (String poa, NotificationListener listener) set Notification Listener with client library.</pre>	22
void	<pre>setProtocol (String protocol) Set using protocol.</pre>	19
void	<pre>setRequestIDCount(long count) set Counter for the RequestID.</pre>	20
void	<pre>setRequestIDHeader(String format) set header for request ID</pre>	20
void	<pre>setSecureProtocol (boolean isSecure) Set security mode for the communication.</pre>	20
void	<pre>setSerialization(String serialization) Set using serialization.</pre>	20
ResourceDT O	<pre>update (String uri, ResourceDTO resource) Update resource</pre>	22

Method Detail

setProtocol

void setProtocol(String protocol)

Set using protocol. This method must be called once before using actual operation methods.

Parameters:

protocol - HTTP, CoAP, MQTT, WebSocket

setSerialization

void setSerialization(String serialization)

Set using serialization. This method must be called once before using actual operation methods.

Parameters:

serialization - using serialization. allowed value is one of "XML", "JSON", "CBOR"

setSecureProtocol

void setSecureProtocol(boolean isSecure)

Set security mode for the communication. This method must be called once before using acutal operation methods.

setAeld

void setAeId(String aeid)

set Application ID. This value is used for registration process of oneM2M. "C" is used for the initial registration. After registration the assigned AEID.

getAeld

String getAeId()

get AEID. This may not differ the parameter used with setAeId() methods. Client Library replace the AEID after oneM2M CSE assigned AEID.

Returns:

Application ID

setRequestIDHeader

void setRequestIDHeader(String format)

set header for request ID

setRequestIDCount

void setRequestIDCount(long count)

set Counter for the RequestID. Usually it is managed by the client library and no need to touch. In case of the special use case, the counter can be manipulated.

getRequestIDCount

```
long getRequestIDCount()
```

get Counter for the RequestID. Usually it is managed by the client library and no need to touch. In case of the special use case, the counter can be manipulated.

Returns:

count

create

Create resource

Parameters:

uri - URI for the target Resource resource - ResourceDTO for creating resource

Returns:

ResourceDTO for created resource

Throws:

OneM2MException

retrieve

```
ResourceDTO retrieve (String uri)
```

Retrieve resource

Parameters:

uri - URI for the target Resource

Returns:

ResourceDTO for retrieved resource

retrieve

Retrieve of partial attributes.

Parameters:

```
uri - URI for the target Resource attributes - attribute names for retrieving attribute.
```

Returns:

ResourceDTO for retrieved resource

Throws:

OneM2MException

update

 $\frac{\text{ResourceDTO}}{\text{ResourceDTO}} \; \begin{array}{c} \textbf{update} \, (\text{String uri,} \\ & \\ & \\ & \\ \text{EsourceDTO} \end{array} \; \text{resource)} \\ \text{throws} \; \frac{\text{OneM2MException}}{\text{MesourceDTO}} \\ \end{array}$

Update resource

Parameters:

uri - URI for resource resource - resource data

Throws:

OneM2MException

delete

void delete(String uri)
 throws OneM2MException

Delete resource

Parameters:

uri - URI for resource

Throws:

OneM2MException

notify

void notify(String poa,

NotificationDTO notification)

throws $\underline{\texttt{OneM2MException}}$

send notification

Throws:

OneM2MException

setNotificationListener

set Notification Listener with client library.

Interface CllientLibraryFactory

org.osgi.onem2m.client

All Superinterfaces:

org.osgi.framework.ServiceFactory<ClientLibrary>

public interface CllientLibraryFactory
extends org.osgi.framework.ServiceFactory<<u>ClientLibrary</u>>

Methods inherited from interface org.osgi.framework.ServiceFactory

getService, ungetService

Interface NotificationListener

org.osgi.onem2m.client

public interface NotificationListener

Interface for Notification Listener White Board Pattern にする場合。 Application wanting to receive notification, MUST register a service implementing this interface, with POA URI in "org.osgi.onem2m.POA" service property. NOTE: This is like White Board Pattern, but there is some limitation. Only the ClientLibrary instance call the method. Client Library must check the registering bundle. (ここが複雑!)

Method	Summary	Pag e
void	<pre>notify(NotificationDTO notification)</pre>	24

Method Detail

notify

void notify (NotificationDTO notification)

Class OneM2MException

org.osgi.onem2m.client

All Implemented Interfaces:

Serializable

Direct Known Subclasses:

ValidationException

public class OneM2MException
extends Exception

Constructor Summary	Pag e
OneM2MException (String string)	25

Constructor Detail

OneM2MException

public OneM2MException(String string)

Package org.osgi.onem2m.dto

Package containing Data Transfer Objects (DTOs).

See:

Description

Class Summa	ary	Page
<u>AttributeDTO</u>	DTO holding attribute information.	27
NotificationDT O	DTO for Notification.	29
OneM2MDTO	DTO containing oneM2M related data.	30
RequestDTO	DTO containing oneM2M request primitive.	31
ResourceDTO	DTO containing oneM2M resource.	32
ResponseDTO	DTO containing oneM2M response primitive.	33

Package org.osgi.onem2m.dto Description

Package containing Data Transfer Objects (DTOs).

Class AttributeDTO

org.osgi.onem2m.dto

public class AttributeDTO
extends OneM2MDTO

DTO holding attribute information.

Field Su	mmary	Pag e
String	attributeName Attribute name in short name for this attribute.	27
int	ResourceType which this attribute belongs to.	27

Fields inherited from class org.osgi.onem2m.dto.OneM2MDTO map

Constructor Summary	Pag e
<pre>AttributeDTO()</pre>	28

Methods inherited from class org.osgi.dto.DTO	
toString	

Field Detail

type

public int type

ResourceType which this attribute belongs to.

attributeName

public String attributeName

Attribute name in short name for this attribute.

Constructor Detail

AttributeDTO

public AttributeDTO()

Class NotificationDTO

org.osgi.onem2m.dto

java.lang.Object
Lorg.osgi.dto.DTO
Lorg.osgi.onem2m.dto.OneM2MDTO
Lorg.osgi.onem2m.dto.NotificationDTO

 $\begin{array}{ll} \text{public class } \textbf{NotificationDTO} \\ \text{extends } \underline{\text{OneM2MDTO}} \end{array}$

DTO for Notification.

Fields inherited from class org.osgi.onem2m.dto.OneM2MDTO

map

Constructor Summary	Pag e	
NotificationDTO()	29	

Methods inherited from cl	lass org.osgi.dto.DTO
toString	

Constructor Detail

NotificationDTO

public NotificationDTO()

Class OneM2MDTO

org.osgi.onem2m.dto

Direct Known Subclasses:

AttributeDTO, NotificationDTO, RequestDTO, ResourceDTO, ResponseDTO

public class OneM2MDTO
extends org.osgi.dto.DTO

DTO containing oneM2M related data.

Field S	ımmary	Pag e	
Map <string< th=""><th>map map holding JSON like data structure</th><th>30</th><th></th></string<>	map map holding JSON like data structure	30	

(Constructor Summary	Pag e
[OneM2MDTO ()	30

Methods inherited from class org.osgi.dto.DTO toString

Field Detail

map

public Map<String,Object> map

map holding JSON like data structure

Constructor Detail

OneM2MDTO

public OneM2MDTO()

Class RequestDTO

org.osgi.onem2m.dto

 $\begin{array}{ll} \text{public class } \textbf{RequestDTO} \\ \text{extends } \underline{\text{OneM2MDTO}} \end{array}$

DTO containing oneM2M request primitive.

Fields inherited from class org.osgi.onem2m.dto.OneM2MDTO

map

Constructor Summary	Pag e
RequestDTO()	31

Methods inherited from class org.osgi.dto.DTO	
toString	

Constructor Detail

RequestDTO

public RequestDTO()

Class ResourceDTO

org.osgi.onem2m.dto

public class ResourceDTO
extends OneM2MDTO

DTO containing oneM2M resource.

Field Su	ımmary	Pag e	
String	resourceType	32	
	resource type	32	

Fields inherited from class org.osgi.onem2m.dto.<u>OneM2MDTO</u>

map

Constructor Summary	Pag e	
ResourceDTO()	32	

Methods inherited from class org.osgi.dto.DTO	
toString	

Field Detail

resourceType

public String resourceType

resource type

Constructor Detail

ResourceDTO

public ResourceDTO()

Class ResponseDTO

org.osgi.onem2m.dto

java.lang.Object
Lorg.osgi.dto.DTO
Lorg.osgi.onem2m.dto.OneM2MDTO
Lorg.osgi.onem2m.dto.ResponseDTO

public class ResponseDTO
extends OneM2MDTO

DTO containing oneM2M response primitive.

Fields inherited from class org.osgi.onem2m.dto.OneM2MDTO

map

Constructor Summary	
ResponseDTO()	33

Methods inherited from class org.osgi.dto.DTO
toString

Constructor Detail

ResponseDTO

public ResponseDTO()

Package org.osgi.onem2m.mapper

Package contains Mapper interface that convert binary data to OneM2MDTO, vice versa.

See:

Description

Interface Summary		Page
<u>Mapper</u>	Mapper interface, which convert OneM2MDTO to binary data and vice versa.	35

Package org.osgi.onem2m.mapper Description

Package contains Mapper interface that convert binary data to OneM2MDTO, vice versa.

Interface Mapper

org.osgi.onem2m.mapper

public interface Mapper

Mapper interface, which convert OneM2MDTO to binary data and vice versa.

Method	Summary	Pag e
OneM2MDTO	<pre>deserialize(byte[] bb)</pre>	35
OneM2MDTO	<pre>deserialize (ByteBuffer bb)</pre>	35
String	<pre>getContentInfo()</pre>	35
ByteBuffer	<pre>serializeToByteBuffer(OneM2MDTO dto)</pre>	35
byte[]	<pre>serializeToBytes (OneM2MDTO dto)</pre>	35

Method Detail

serializeToByteBuffer

ByteBuffer serializeToByteBuffer(OneM2MDTO dto)

serializeToBytes

byte[] serializeToBytes(OneM2MDTO dto)

deserialize

OneM2MDTO deserialize(ByteBuffer bb)

deserialize

OneM2MDTO deserialize(byte[] bb)

getContentInfo

String getContentInfo()

Package org.osgi.onem2m.servicelayer

The key Package containing Service Layer API.

See:

Description

Interface Sum	ımary	Page
<u>Ae</u>	Ae, which represents oneM2M AE.	37
<u>Cse</u>	CSEService, which represents oneM2M CSE.	37
ProtocolBindin g	Protocol Binding Service, which represent Protocol Binding	38
ServiceLayer	Service Layer Interface, which locates between CSE and Protocol Binding Service.	39

Package org.osgi.onem2m.servicelayer Description

The key Package containing Service Layer API.

Interface Ae

org.osgi.onem2m.servicelayer

All Superinterfaces:

ServiceLayer

public interface Ae
extends ServiceLayer

Ae, which represents oneM2M AE.

 $Methods\ inherited\ from\ interface\ org.osgi.onem 2m. service layer. \underline{Service Layer}$

request

Interface Cse

org.osgi.onem2m.servicelayer

All Superinterfaces:

ServiceLayer

public interface Cse
extends ServiceLayer

CSEService, which represents oneM2M CSE.

Methods inherited from interface org.osgi.onem2m.servicelayer.ServiceLayer

request

Interface ProtocolBinding

org.osgi.onem2m.servicelayer

All Superinterfaces:

ServiceLayer

 $\begin{array}{ll} \text{public interface } \textbf{ProtocolBinding} \\ \text{extends } \underline{\textbf{ServiceLayer}} \end{array}$

Protocol Binding Service, which represent Protocol Binding

Methods inherited from interface org.osgi.onem2m.servicelayer.ServiceLayer

request

Interface ServiceLayer

org.osgi.onem2m.servicelayer

All Known Subinterfaces:

Ae, Cse, ProtocolBinding

public interface ServiceLayer

Service Layer Interface, which locates between CSE and Protocol Binding Service.

Method	Summary	Pag e
org.osgi.u til.promis	<pre>request(RequestDTO request)</pre>	
e.Promise<		40
ResponseDT O>		

Method Detail

request

org.osgi.util.promise.Promise<<u>ResponseDTO</u>> request(<u>RequestDTO</u> request)

send a request.

Parameters:

request - request

Returns:

promise for ResoponseDTO.

Package org.osgi.onem2m.util

Package containing Utility Classes for oneM2M.

See:

Description

Class Summa	Class Summary	
<u>Util</u>	Utility Class for oneM2M.	42

Package org.osgi.onem2m.util Description

Package containing Utility Classes for oneM2M.

Class Util

org.osgi.onem2m.util

java.lang.Object

└org.osgi.onem2m.util.Util

public class Util
extends Object

Utility Class for oneM2M.

C	Constructor Summary	Pag e
U	<u>Jtil</u> ()	42

Method	Method Summary	
AttributeD TO	<pre>getAttributeS (ResourceDTO resource, String sname) retrieve AttributeDTO from ResourceDTO using short named Attribute</pre>	43
String	12s (String lname) Convert long name to short name.	43
static void	<pre>main (String[] args) Test purpose main() function.</pre>	44
OneM2MDTO	<pre>put (OneM2MDTO dto, String key, int value) Put data into a map managed in OneM2MDTO.</pre>	43
OneM2MDTO	<pre>put(OneM2MDTO dto, String key, Object value) Put data into a map managed in OneM2MDTO.</pre>	43
OneM2MDTO	<pre>put (OneM2MDTO dto, String key, long value) Put data into a map managed in OneM2MDTO.</pre>	44
String	<pre>s21 (String sname) Convert short name to long name.</pre>	42

Constructor Detail

Util

public Util()

Method Detail

s2l

public String s21(String sname)

Convert short name to long name.

Parameters:

sname - short name

Returns:

long name

I2s

```
public String 12s(String lname)
```

Convert long name to short name.

Parameters:

lname - long name

Returns:

short name

getAttributeS

```
 \begin{array}{ccc} \text{public } \underline{\text{AttributeDTO}} & \textbf{getAttributeS} (\underline{\text{ResourceDTO}} & \text{resource,} \\ & \text{String sname)} \end{array}
```

retrieve AttributeDTO from ResourceDTO using short named Attribute

Parameters:

resource - ResourceDTO sname - Attribute name in short name

Returns:

AttributeDTO

put

Put data into a map managed in OneM2MDTO.

Parameters:

dto - OneM2MDTO key - key value - value

Returns:

OneMDMDTO specified as the first argument, allowing method chaining.

put

Put data into a map managed in OneM2MDTO.

Parameters:

dto - OneM2MDTO

```
key - key value - value
```

Returns

OneMDMDTO specified as the first argument, allowing method chaining.

put

Put data into a map managed in OneM2MDTO.

Parameters:

dto - OneM2MDTO key - key value - value

Returns:

OneMDMDTO specified as the first argument, allowing method chaining.

main

```
public static void main(String[] args)
```

Test purpose main() function.

Package org.osgi.onem2m.validation

Package containing validation interface for oneM2M data structures.

See:

Description

Interface Summary		Page
RequestValidat or	Resource Validator interface.	46
ResourceValida tor	Resource Validator interface.	47
ResponseValid ator	Response Validator interface.	48

Exception Summary		Page
ValidationExce ption	Validation Exception, which will be thrown in Validation related processing.	49

Package org.osgi.onem2m.validation Description

Package containing validation interface for oneM2M data structures.

Interface RequestValidator

org.osgi.onem2m.validation

public interface RequestValidator

Resource Validator interface.

Method Summary	,	Pag e
void validate (RequestDTO req)		46
validate RequestDTO		40

Method Detail

validate

void **validate**(RequestDTO req)
throws ValidationException

validate RequestDTO

Throws:

<u>ValidationException</u>

Interface ResourceValidator

org.osgi.onem2m.validation

public interface ResourceValidator

Resource Validator interface.

Method Summary	F	Pag e
void validate (ResourceDTO resource)		47
Validate ResourceDTO	'	41

Method Detail

validate

void **validate**(<u>ResourceDTO</u> resource) throws <u>ValidationException</u>

Validate ResourceDTO

Parameters:

resource - resourceDTO under validation.

Throws:

 $\underline{\tt ValidationException}$

Interface ResponseValidator

org.osgi.onem2m.validation

public interface ResponseValidator

Response Validator interface.

Method	Summary	Pag e	
void	<pre>validate(ResponseDTO response)</pre>	48	
	Response Validator.	40	

Method Detail

validate

void **validate**(<u>ResponseDTO</u> response) throws <u>ValidationException</u>

Response Validator.

Parameters:

response - ResoposeDTO under validation

Throws:

<u>ValidationException</u>

Class ValidationException

org.osgi.onem2m.validation

All Implemented Interfaces:

Serializable

public class ValidationException
extends OneM2MException

Validation Exception, which will be thrown in Validation related processing.

Constructor Summary	Pag e	
<pre>ValidationException (String string)</pre>	49	Ī

Constructor Detail

ValidationException

public ValidationException (String string)

Java API documentation generated with DocFlex/Doclet v1.6.1

DocFlex/Doclet is both a multi-format Javadoc doclet and a free edition of DocFlex/Javadoc. If you need to customize your Javadoc without writing a full-blown doclet from scratch, DocFlex/Javadoc may be the only tool able to help you! Find out more at www.docflex.com

8 Considered Alternatives

For posterity, record the design alternatives that were considered but rejected along with the reason for rejection. This is especially important for external/earlier solutions that were deemed not applicable.

8.1 Representation of DTO

As alternative solution, utilization of generated Java classes by JAXB has been considered, since oneM2M provides well defined XSD for defining data format. With the following aspects, this approach is not applied.

Many classes: Currently 65 XSD files are defined in oneM2M specification and JAXB tool (xjc) generates more than 140 Java classes. Using many classes as interface could make specification more complicated than its nature.

No Uniqueness: Generated classes by xjc are not unique, because it is possible to customize generation processes.

Changeability: Depending on the version of oneM2M, XSD files differ. It is preferable to choose version independent API, as much as possible. oneM2M ensures any data can be converted to JSON and CBOR, so proposed approach can be used with out modification, even if XSD file would be changed.

8.2 White Board pattern for receiving notification by AE

For setNotificationListener() methods on ClientLibrary interface, the alternative using White Board pattern is considered, in which the listener is registered on to the OSGi Service Registry, instead of calling setter method. But the listener should be called from only corresponding ClientLibrary Service instance, so the current design was chosen.

8.3 Non blocking API for ClientLibrary

Currently ClientLibrary is desinged as blocking concept. There was an option to design as non blocking API. Because developers want to create AE using non blocking API, she or he can create directly on top of ServiceLayer API. So Non blocking API for ClientLibrary is not provided.

9 Security Considerations

Description of all known vulnerabilities this may either introduce or address as well as scenarios of how the weaknesses could be circumvented.

9.1 ProtocolBinding Service with secure protocols

In case that ProtocolBInding Service uses secure protocols, it is expected to handle pre-shared key or certificate, in order to get authorized by the communication peer. Once Ae gets the service, it can use it to communicate. The ProtocolBinding service should be protected, in the environment that accommodates many different bundles from different vendors/providers.

9.2 Using Multiple Certificates with in a single ProtocolBinidng Service

With higher degree of aggregation of bundles from different vendos/providers on a single OSGi framework, it would be beneficial for ProtocolBinding service would handle multiple certificates through API. But it is not addressed yet, so far. It might be out of scope of this RFC.

10 Document Support

10.1 References

- [1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.
- [2]. oneM2M TS-0001 Functional Architecture, http://onem2m.org/images/files/deliverables/Release2/TS-

0001-%20Functional_Architecture-V2_10_0.pdf

- [3]. oneM2M TS-0004 Service Layer Core Protocol, http://onem2m.org/images/files/deliverables/Release2/TS-0004 Service Layer Core Protocol V2 7 1.zip
- [4]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0 (NOTE:Is this needed?)

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.

10.2 Author's Address

Name	Hiroyuki Maeomichi
Company	NTT
Address	Midorimachi 3-9-11, Musashino, Tokyo, Japan
Voice	+81 422 59 4072
e-mail	maeomichi.hiroyuki@lab.ntt.co.jp

10.3 Acronyms and Abbreviations

CSE: Common Services Entity

AE: Application Entity

CBOR: Concise Binary Object Representation

10.4 End of Document