Service Layer API for oneM2M

Draft

35 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,

worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future ("Future Specification"), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design
The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 10.1.

`Source code is shown in this typeface.`

0.6 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	*SEP 15 2017*	Initial Contribution. Hiroyuki Maeomichi, NTT, maeomichi.hiroyuki@lab.ntt.co.jp
0.0.1	*SEP 21 2017*	Updated alternatives, some figures, added description on validator. Hiroyuki Maeomichi, NTT, maeomichi.hiroyuki@lab.ntt.co.jp

Revision	Date	Comments
0.02	*April 17 2018*	Update based on discussion in Washington meeting. Hiroyuki Maeomichi, NTT, maeomichi.hiroyuki@lab.ntt.co.jp

1 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).

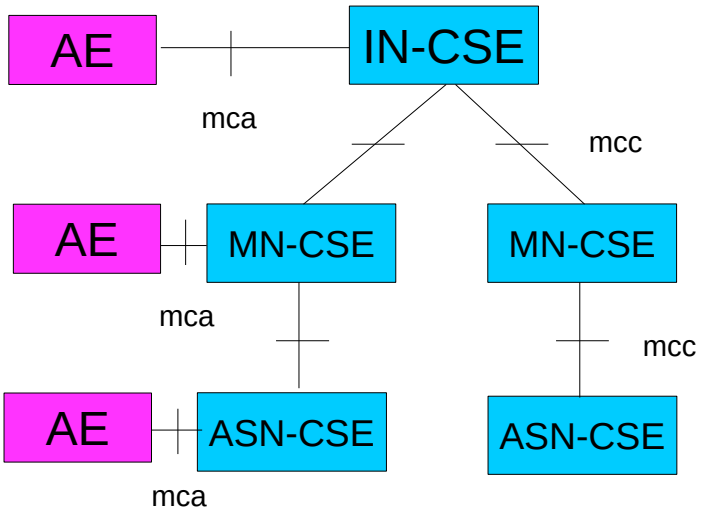
This RFP discuss the way to provide high level API (namely service layer API) for oneM2M RESTful operations hiding the difference of variety of communication methods.

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.

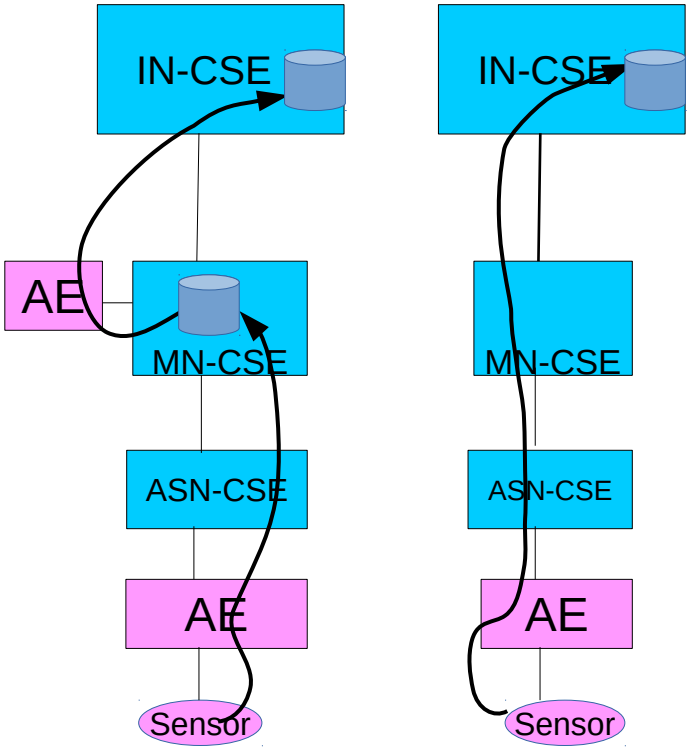


oneM2M's application, called Application Entity (AE) connects to one of CSEs. After AE connecting to the CSE, AE can access to all of CSEs, by retargeting function (similar to routing) of CSEs.

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 <c*ontainer*> 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 *mc*a*,* 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.

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)

- 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.

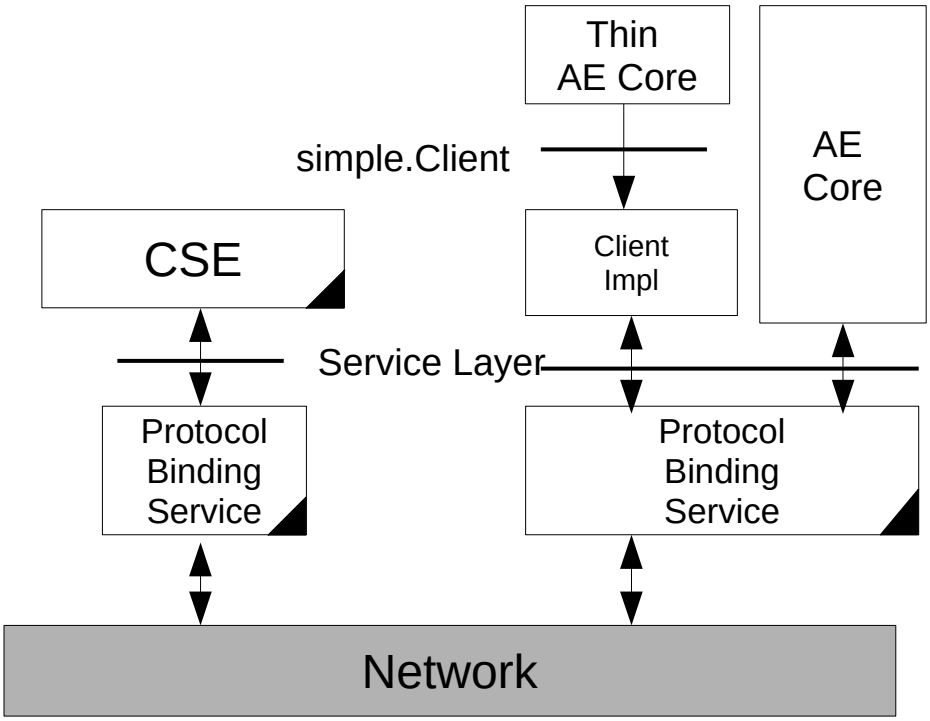
Strictly use the terminology a defined in the Problem Context.

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.



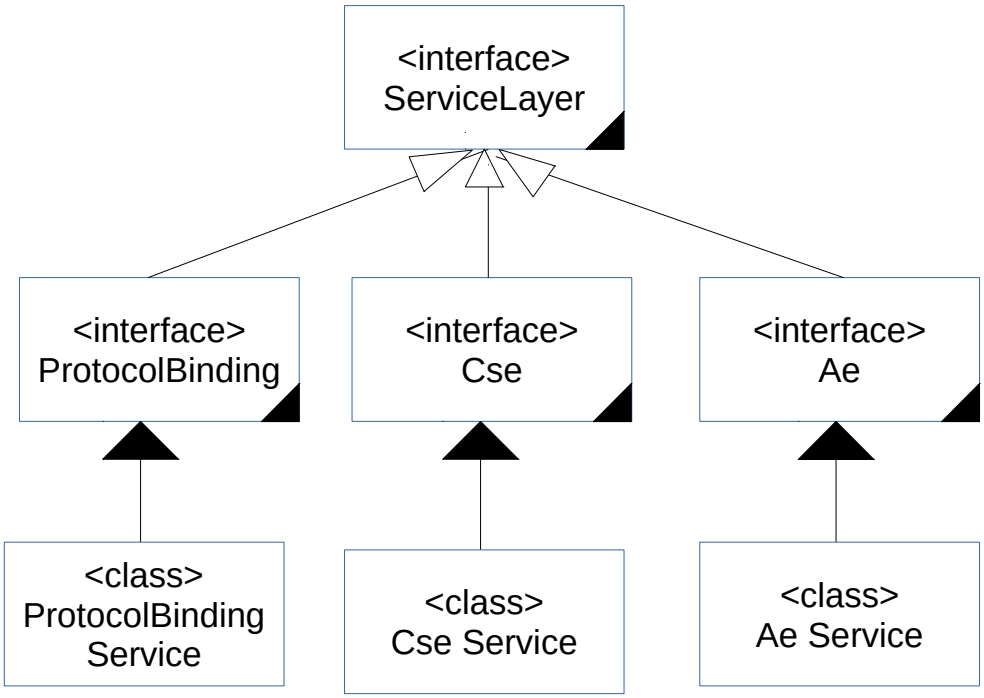
Service Layer Interface is defined as follows. Only method request sends request message and return Promise for the response. Here, Promise enables asynchronous messaging.

```
package org.osgi.onem2m.servicelayer;

import org.osgi.onem2m.dto.RequestDTO;
import org.osgi.onem2m.dto.ResponseDTO;
import org.osgi.util.promise.Promise;

/**
 * Service Layer Interface, which locates between CSE and Protocol Binding Service.
 */
public interface ServiceLayer {
	/**
	 * send a request.
	 *
	 * @param request request
	 * @return promise for ResponseDTO.
```

```
        */
        Promise<ResponseDTO> request(RequestDTO request);
}
```

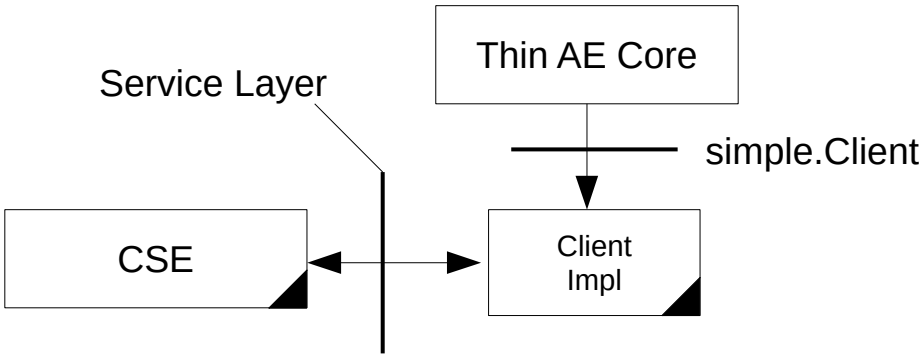
On the Service Layer interface, there are bidirectional invocations, that is, CSE sends request to protocol binding service and get response from it, protocol binding service sends request to CSE and get response from it. For the clarification purpose, Service Layer Interface is extended to ProtocolBinding 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	org.osgi.service.onem2m .ProtocolBinding	Supporting protocol.
	serialization	org.osgi.service.onem2m .Serialization	Serialization.
	secure	boolean	True, if secure protocol is supported, otherwise false.
	versions	org.osgi.service.onem2m .SpecificationVersion[]	Supported versions.
	forAPP-ID	String	Indicates which Application can use this service.
Cse	CSE-ID	String	CSE-ID: ID of CSE
	SP-ID	String	ID of Service Provider
	CSE-type	org.osgi.service.onem2m .CSEType	Type of CSE. Possible values are IN, MN, or ASN
	POA	String[]	URI for point of access
	versions	org.osgi.service.onem2m .SpecificationVersion[]	Supported versions
Ae	AE-ID	String	ID of Application Entity
	APP-ID	String	Application ID
	POA	String[]	URI for point of access
	versions	org.osgi.service.onem2m .SpecificationVersion[]	Supported versions

5.4 Introspection Interface

5.4.1 General Introspection Interface

This interface checks validity of Data structure. For oneM2M resources there are 3 types of attributes, which are mandatory, optional and NP(Not present) and they are different between create operation and update operation. So there are 6 variations of methods for getting attribute names.

```
package org.osgi.service.onem2m;
import org.osgi.service.onem2m.dto.*;

public interface Introspector {

	/**
	 * execute Validation of Data Structure
	 *
	 * @param resp
	 * @return array of problems
	 */
	public String[] findValidationProblems(ResponsePrimitiveDTO resp);

	/**
	 * execute Validation of Data Structure
	 *
	 * @param resp
	 * @return array of problems
	 */
	public String[] findValidationProblems(RequestPrimitiveDTO req);

	/**
	 * execute Validation of Data Structure
	 *
	 * @param resp
	 * @return array of problems
	 */
	public String[] findValidationProblems(ResourceDTO resource);

	/**
	 * get Possible Attribute Names for the given resouceType.
	 *
	 * @param resp
	 * @return array of Possible Attributes
	 */
	public String[] getAttributeNames(int resourceType);

	/**
	 * get Mandatory Attributes for the given resouceType when Create.
	 *
```

```
     * @param resp
     * @return array of Possible Attributes
     */
    public String[] getMandatoryAttributesForCreate(int resourceType);

    /**
     * get Optional Attributes for the given resouceType when Create.
     *
     * @param resp
     * @return array of Possible Attributes
     */
    public String[] getOptionalAttributesForCreate(int resourceType);

    /**
     * get Not Present Attributes for the given resouceType when Create.
     *
     * @param resp
     * @return array of Possible Attributes
     */
    public String[] getNotPresentAttributesForCreate(int resourceType);


    /**
     * get Mandatory Attributes for the given resouceType when Update.
     *
     * @param resp
     * @return array of Possible Attributes
     */
    public String[] getMandatoryAttributesForUpdate(int resourceType);

    /**
     * get Optional Attributes for the given resourceType when Update.
     *
     * @param resp
     * @return array of Possible Attributes
     */
    public String[] getOptionalAttributesForUpdate(int resourceType);

    /**
     * get Not Present Attributes for the given resouceType when Update.
     *
     * @param resp
     * @return array of Possible Attributes
     */
    public String[] getNotPresentAttributesForUpdate(int resourceType);

    /**
```

```
	 * return Java Type for given attribute of resource type
	 * @param resouceType
	 * @param attribute
	 * @return expected class for the specified attribute
	 */
	public Class getType(int resouceType, String Attribute);

	/**
	 * return Typical Data Structure for Type for given attribute of resource type
	 * @param resouceType
	 * @param attribute
	 * @return Template Object
	 */
	public Object getTemplateObject(int resouceType, String Attribute);

}
```

5.4.2 Introspection interface for FlexContainer

FlexContainer Resource can be defined by user with adding custom attributes. Initially it was intended to be like JSON structure in oneM2M world. The definition of custom flex container is expressed in 'contentDefinition' attribute of the FlexContainer. It include URI of XSD definition; The XSD content is stored as contentInstance resource in some CSE.

This interface checks validity of data structure representing <FlexContainer> resource. Following table shows service properties on the service.

Property Name	Type	explanation
contentDefinitions	String[]	Supporting contentDefinition of <FlexContainer>. .

```
package org.osgi.service.onem2m;

import org.osgi.service.onem2m.dto.*;

/**
 * FlexContainerInspector
 *
 *
 */
public interface FlexContainerIntrospector {

	/**
	 * Execute Validation of Data Structure
	 *
	 * @param resp
	 * @return array of problems
	 */
	public String[] findValidationProblems(ResourceDTO resource);
```

```
	/**
	 * get Possible Attributes for the given resouceType.
	 *
	 * @param resp
	 * @return array of Possible Attributes
	 */
	public String[] getCustomeAttributeNames(String containerDefinition);

	/**
	 * return Java Type for given attribute of resource type
	 *
	 * @param resouceType
	 * @param attribute
	 * @return expected class for the specified attribute
	 */
	public Class getType(String containerDefinition, String customAttributeName);

	/**
	 * return Typical Data Structure for Type for given attribute of resource type
	 *
	 * @param resouceType
	 * @param attribute
	 * @return Template Object
	 */
	public Object getTemplateObject(String containerDefinition, String
customAttributeName);

}
```

5.5 Client library

Client Interface provides more high level api to application. It 1) provides CRUD operations granularity, hiding detail data structure of RequestPrimitive and ResponsePrimitive, and, 2) automatically assigns some protocol parameters like request ids or from.

```
package org.osgi.service.onem2m.simple;
import org.osgi.service.onem2m.dto.*;
import org.osgi.service.onem2m.*;
import java.util.*;

public interface Client {
	/**
	 * create resource
	 *
	 * @param uri URI for parent resource
```

```
         * @param resource resource data
         * @return created resource
         * @throws OneM2MException
         */
        public ResourceDTO create(String uri, ResourceDTO resource) throws
OneM2MException;

        /**
         * retrieve resource
         *
         * @param uri uri for retrieving resource
         * @return retrieved resource data
         * @throws OneM2MException
         */
        public ResourceDTO retrieve(String uri) throws OneM2MException;

        /**
         * retrieve subset of attributes.
         *
         * @param uri uri for retrieving resource
         * @param nesessaryAttributeNames attribute names for retrival
         * @return retrieved resource data
         * @throws OneM2MException
         */
        public ResourceDTO retrieve(String uri, String[] nesessaryAttributeNames)
throws OneM2MException;

        /**
         * update resource
         *
         * @param uri uri for updating resource
         * @param resource data resource
         * @return updated resource
         * @throws OneM2MException
         */
        public ResourceDTO update(String uri, ResourceDTO resource) throws
OneM2MException;

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

        /**
```

```
	 * find resources
	 *
	 * @param uri uri for top of search
	 * @param fc filter criteria
	 * @return list of URIs maching the condition specified in fc
	 * @throws OneM2MException
	 */
	public List<String> discover(String uri, FilterCriteriaDTO fc) throws
OneM2MException;

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

package org.osgi.service.onem2m.simple.impl;

public class ClientImpl implements Client {
	public ClientImpl(String appid, String aeid, NotificationListener listener) {}

	public ClientImpl(String appid, String aeid, NotificationListener listener,
Client customizer) {}
}
```

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.

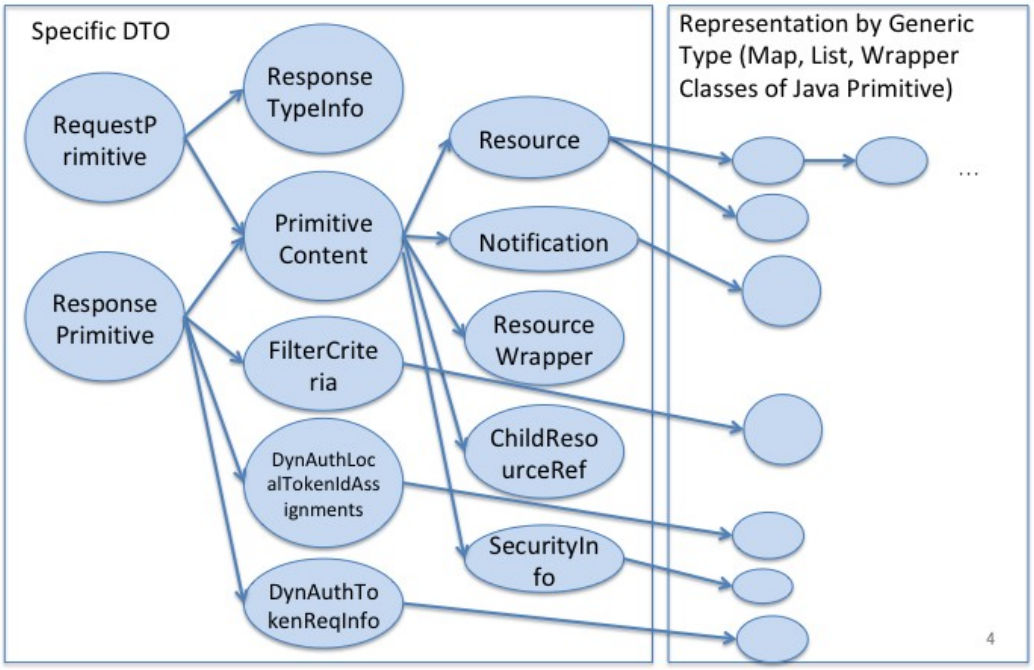
For all new functionality added to the OSGi Framework the question should be asked: would this feature benefit from a DTO? The expectation is that in most cases it would.

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 DTO

Data transfer Object was chosen following OSGi's designing manner. But the data structure near root structures are designed specific DTO and other resources is to be stored as generic types, such as Map, List and Wrapper classes of Java Primitives. (See also the considered Alternatives)




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

import java.util.Map;
import org.osgi.dto.DTO;

/**
 * DTO containing oneM2M related data.
 *
 */
public class OneM2MDTO extends DTO {
	/**
	 * map holding JSON like data structure
	 */
	public Map<String,Object> map;
}
```

6.2 RequestPrimitiveDTO

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

import java.util.*;

public class RequestPrimitiveDTO extends org.osgi.dto.DTO {
	@javax.xml.bind.annotation.XmlElement(required = true)
```

```
	public Operation operation;
	@javax.xml.bind.annotation.XmlElement(required = true)
	public String to;
	public String from;
	@javax.xml.bind.annotation.XmlElement(required = true)
	public String requestIdentifier;
	@javax.xml.bind.annotation.XmlElement(required = false)
	public Integer resourceType;
	public PrimitiveContentDTO primitiveContent;
	public List<String> roleIDs;
	public String originatingTimestamp;
	public String requestExpirationTimestamp;
	public String resultExpirationTimestamp;
	public String operationExecutionTime;
	public ResponseTypeInfoDTO responseType;
	public String resultPersistence;
	@javax.xml.bind.annotation.XmlElement(required = false)
	public ResultContent resultContent;
	public String eventCategory;
	@javax.xml.bind.annotation.XmlElement(required = false)
	public Boolean deliveryAggregation;
	public String groupRequestIdentifier;
	public FilterCriteriaDTO filterCriteria;
	@javax.xml.bind.annotation.XmlElement(required = false)
	public DiscoveryResultType discoveryResultType;
	public String tokens;
	public String tokenIDs;
	public List<String> localTokenIDs;
	@javax.xml.bind.annotation.XmlElement(required = false)
	public Boolean tokenReqIndicator;

	public static enum DiscoveryResultType {
		structured(1), unstructured(2);

		int value;

		private DiscoveryResultType(int i) {
			value = i;
		}

		public int getValue() {
			return value;
		}
	}

	public static enum ResultContent {
```

```
		nothing(1), attributes(2), hierarchicalAddress(3),
	hierarchicalAddressAndAttributes(4),
	attributesAndChildResources(5),
	attributesAndChildResourceReferences(6),
	childResourceReferences(7), originalResource(8), childResousrces(9);

		int value;

		private ResultContent(int i) {
			value = i;
		}

		public int getValue() {
			return value;
		}
	}

	public static enum Operation {
		Create(1), Retrieve(2), Update(3), Delete(4), Notify(5);

		int value;

		private Operation(int i) {
			value = i;
		}

		public int getValue() {
			return value;
		}
	}

}
```

6.3 ResponsePrimitiveDTO

ResponsePrimitiveDTO holds a Response Information used for oneM2M communication.

```
package org.osgi.service.onem2m.dto;
import java.util.*;

public class ResponsePrimitiveDTO extends org.osgi.dto.DTO{
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public Integer responseStatusCode;
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public String requestIdentifier;
```

```
	public PrimitiveContentDTO primitiveContent;
	public String to;
	public String from;
	public String originatingTimestamp;
	public String resultExpirationTimestamp;
	public String eventCategory;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer contentStatus;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer contentOffset;
	public DynAuthLocalTokenIdAssignmentsDTO assignedTokenIdentifiers;
//Map<String,Object>
	public DynAuthTokenReqInfoDTO tokenReqInfo;//DynAuthTokenReqInfoDTO
}
```

6.4 ResponseTypeInfoDTO

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

import java.util.*;

public class ResponseTypeInfoDTO extends org.osgi.dto.DTO {
	@javax.xml.bind.annotation.XmlElement(required = true)
	public java.lang.Integer responseTypeValue;
	@javax.xml.bind.annotation.XmlElement(required = true)
	public List<java.lang.String> notificationURI;

	public static enum ResponseType {
		nonBlockingRequestSynch(1), nonBlockingRequestAsynch(2),
blockingRequest(3), flexBlocking(4);

		private int value;

		private ResponseType(int i) {
			value = i;
		}

		public int getValue() {
			return value;
		}
	}
}
```

6.5 FilterCriteriaDTO

```
package org.osgi.service.onem2m.dto;
import java.util.*;

public class FilterCriteriaDTO extends org.osgi.dto.DTO{
	public String createdBefore;
	public String createdAfter;
	public String modifiedSince;
	public String unmodifiedSince;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer stateTagSmaller;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer stateTagBigger;
	public String expireBefore;
	public String expireAfter;
	public List<String> labels;
	public List<Integer> resourceType;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer sizeAbove;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer sizeBelow;
	public List<String> contentType;
	public List<Map<String,Object>> attribute;// List<Attribute>
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public FilterUsage filterUsage;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer limit;
	public List<String> semanticsFilter;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public FilterOperation filterOperation;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer contentFilterSyntax;
	public String contentFilterQuery;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer level;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer offset;

	public static enum FilterOperation {
		AND(1), OR(2);

		private int value;

		private FilterOperation(int i) {
			value = i;
```

```
		}

		public int getValue() {
			return value;
		}
	}

	public static enum FilterUsage {
		DiscoveryCriteria(1), ConditionalRetrival(2), IPEOndemandDiscovery(3);

		private int value;

		private FilterUsage(int i) {
			value = i;
		}

		public int getValue() {
			return value;
		}

	}
}
```

6.6 DynAuthLocalTokenIdAssignmentsDTO

```
package org.osgi.service.onem2m.dto;
import java.util.*;
public class DynAuthLocalTokenIdAssignmentsDTO extends org.osgi.dto.DTO{
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public List<Map> localTokenIdAssignment;//LocalTokenIdAssignmentDTO
}
```

6.7 DynAuthTokenReqInfoDTO

```
package org.osgi.service.onem2m.dto;
import java.util.*;
public class DynAuthTokenReqInfoDTO extends org.osgi.dto.DTO{
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public List<Map> dasInfo;//Map : DasInfo
}
```

6.8 ResourceDTO

```
package org.osgi.service.onem2m.dto;
import java.util.*;

public class ResourceDTO extends org.osgi.dto.DTO{

	// Universal Attribute, which can be held by all resources.
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public Integer resourceType;
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public String resourceID;
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public String parentID;
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public String creationTime;
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public String lastModifiedTime;

	public String resourceName;

	// optional, Universal Attributes
	public List<String> labels;

	/**
	 * Non Universal Attribute.
	 * Value Part must be the types that are allowed for OSGi DTO.
	 */
	public Map<String, Object> attribute;
}
```

6.9 NotificationDTO

```
package org.osgi.service.onem2m.dto;
import java.util.*;

public class NotificationDTO extends org.osgi.dto.DTO{
	public Map<String,Object> notificationEvent;//NotificationEventDTO
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Boolean verificationRequest;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Boolean subscriptionDeletion;
```

```
	public String subscriptionReference;
	public String creator;
	public String notificationForwardingURI;
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Map<String,Object> ipeDiscoveryRequest;//IPEDiscoveryRequestDTO
}
```

6.10 ResourceWrapperDTO

```
package org.osgi.service.onem2m.dto;
public class ResourceWrapperDTO extends org.osgi.dto.DTO{
	@javax.xml.bind.annotation.XmlElement( required  = true)
	public String uri;

	ResourceDTO resource;
}
```

6.11 ChildResourceRefDTO

```
package org.osgi.service.onem2m.dto;
public class ChildResourceRefDTO extends org.osgi.dto.DTO{
	public String uri;
	public String name;
	public Integer type;//Resource Type    TODO: should be int ?
	public String specializationID;// any URI, optional
}
```

6.12 SecurityInfoDTO

```
package org.osgi.service.onem2m.dto;
import java.util.*;
public class SecurityInfoDTO extends org.osgi.dto.DTO{
	@javax.xml.bind.annotation.XmlElement( required  = false)
	public Integer securityInfoType;
	public Map<String,Object>  dasRequest;// DynAuthDasRequestDTO
	public Map<String,Object>  dasResponse;//DynAuthDasResponseDTO
	public Map<String,Object> esprimRandObject;//ReceiverESPrimRandObjectDTO
	public String esprimObject;
	public byte[] escertkeMessage;
}
```

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

Demo Documentation

18/04/17 17:46

Package Summary		Page
org.osgi.service.onem2m.simple.impl		28

Package org.osgi.service.onem2m.simple.impl

Class Summary		Page
ClientImpl		29

Class ClientImpl

org.osgi.service.onem2m.simple.impl

```
java.lang.Object
  └ org.osgi.service.onem2m.simple.impl.ClientImpl
```

All Implemented Interfaces:

org.osgi.service.onem2m.simple.Client

```
public class ClientImpl
extends Object
implements org.osgi.service.onem2m.simple.Client
```

Constructor Summary	Page
`ClientImpl(String appid, String aeidstem, org.osgi.service.onem2m.simple.NotificationListener listener)`	30
`ClientImpl(String appid, String aeidstem, org.osgi.service.onem2m.simple.NotificationListener listener, org.osgi.service.onem2m.simple.Client customizer)`	31

Method Summary		Page
org.osgi.service.onem2m.dto.ResourceDTO	`create(String uri, org.osgi.service.onem2m.dto.ResourceDTO resource)` create resource	31
void	`delete(String uri)` delete resource	32
List	`discover(String uri, org.osgi.service.onem2m.dto.FilterCriteriaDTO fc)` find resources	32
void	`notify(org.osgi.service.onem2m.dto.NotificationDTO notification)` send notification	32
org.osgi.service.onem2m.dto.ResourceDTO	`retrieve(String uri)` retrieve resource	31
org.osgi.service.onem2m.dto.ResourceDTO	`retrieve(String uri, String[] nesessaryAttributeNames)` retrieve subset of attributes.	31
org.osgi.service.onem2m.dto.ResourceDTO	`update(String uri, org.osgi.service.onem2m.dto.ResourceDTO resource)` update resource	32

Constructor Detail

ClientImpl

```
public ClientImpl(String appid,
                  String aeidstem,
                  org.osgi.service.onem2m.simple.NotificationListener listener)
```

ClientImpl

```
public ClientImpl(String appid,
                  String aeidstem,
                  org.osgi.service.onem2m.simple.NotificationListener listener,
                  org.osgi.service.onem2m.simple.Client customizer)
```

Method Detail

create

```
public org.osgi.service.onem2m.dto.ResourceDTO create(String uri,
                                                      org.osgi.service.onem2m.dto.ResourceDTO
resource)
```

Description copied from interface: **`org.osgi.service.onem2m.simple.Client`**
create resource

Specified by:
`create` in interface `org.osgi.service.onem2m.simple.Client`
Parameters:
`uri` - URI for parent resource
`resource` - resource data
Returns:
created resource

---

retrieve

```
public org.osgi.service.onem2m.dto.ResourceDTO retrieve(String uri)
```

Description copied from interface: **`org.osgi.service.onem2m.simple.Client`**
retrieve resource

Specified by:
`retrieve` in interface `org.osgi.service.onem2m.simple.Client`
Parameters:
`uri` - uri for retrieving resource
Returns:
retrieved resource data

---

retrieve

```
public org.osgi.service.onem2m.dto.ResourceDTO retrieve(String uri,
                                                        String[] nesessaryAttributeNames)
```

Description copied from interface: **`org.osgi.service.onem2m.simple.Client`**
retrieve subset of attributes.

Specified by:
`retrieve` in interface `org.osgi.service.onem2m.simple.Client`
Parameters:
`uri` - uri for retrieving resource
`nesessaryAttributeNames` - attribute names for retrival
Returns:
retrieved resource data

---

update

```
public org.osgi.service.onem2m.dto.ResourceDTO update(String uri,
                                                      org.osgi.service.onem2m.dto.ResourceDTO
resource)
```

Description copied from interface: **`org.osgi.service.onem2m.simple.Client`**
update resource

Specified by:
`update` in interface `org.osgi.service.onem2m.simple.Client`
Parameters:
`uri` - uri for updating resource
`resource` - data resource
Returns:
updated resource

---

delete

```
public void delete(String uri)
```

Description copied from interface: **`org.osgi.service.onem2m.simple.Client`**
delete resource

Specified by:
`delete` in interface `org.osgi.service.onem2m.simple.Client`
Parameters:
`uri` - target uri for deleting resource

---

notify

```
public void notify(org.osgi.service.onem2m.dto.NotificationDTO notification)
```

Description copied from interface: **`org.osgi.service.onem2m.simple.Client`**
send notification

Specified by:
`notify` in interface `org.osgi.service.onem2m.simple.Client`

---

discover

```
public List discover(String uri,
                     org.osgi.service.onem2m.dto.FilterCriteriaDTO fc)
              throws org.osgi.service.onem2m.OneM2MException
```

Description copied from interface: **`org.osgi.service.onem2m.simple.Client`**
find resources

Specified by:
`discover` in interface `org.osgi.service.onem2m.simple.Client`
Parameters:
`uri` - uri for top of search
`fc` - filter criteria

Returns:
list of URIs maching the condition specified in fc
Throws:
`org.osgi.service.onem2m.OneM2MException`

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

8.1.1 JAXB generated Class

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.

8.1.2 GenericDTO

GenricDTO, which has Map<String, Obj> in the top, has been discussed in Gent meeting. But it seems bad usage of defining DTO.

8.1.3 SpecificDTO

SpecificDTO definitions have been generated from XSD generated classes. The number of DTO exceeds 170 and Java doc pages are getting 300 pages. It is apparently too much to express data formats. So middle approach of generic DTO and specific DTO has been chosen.

8.2 Resource Types Expression

In DTO, enum was eagerly used for clear candidates of possible values. But resource types seems more fragile because new resource types could be easily added. So Integer was chosen for resource types.

8.3 Use of Annotation defined by JAXB in DTO

Currently annotations defined by JAXB was used in DTO. It was pointed out as confusing because it might give impression that it only support XML serialization. But it was kept in the definitions by following reasons.

1. Reming the annotations are easier than inserting.

2. It is informative to specify optionality.

New OSGI annotation specifying optionality could be possible, but it might take time because it should be published as Core specification and R7 just has released.

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. If unexpected AE uses the protocol binding services, it causes security hazard. So the ProtocolBinding Service should be protected by permissions. org.osgi.service.onem2m.AppidPermission(applicationid, USE) should be assigned to the AE bundles, and ProtocolBinding service check the permission in method call. (Want to control with ServicePermission!!)

9.2 Using Cse from AE

In case that AE and CSE locates in the same framework, the access from AE to CSE should be managed by ServicePermission(“org.osgi.service.onem2m.CsePermission, “GET”).

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