

ATIS OS-IoT

Manual

May 15th, 2018
(Version 0.5.0 Release)

Iain Sharp,
Principal Technologist, ATIS

Table of Contents

Table of Contents	2
1. Introduction	3
2. Technical Feature Overview	12
3. API Specification	14
4. Protocol Compliance Statement	16
Appendix A – Additional References	28
Appendix B – Compatibility Notes with CSE Implementations	29
Appendix C – Resource Support Reported by Tested CSEs	31

1. Introduction

ATIS OS-IoT is an open source toolkit to simplify the development of IoT clients that are compatible with the oneM2M standard. OS-IoT provides functions to allow clients to implement a oneM2M Application Entity (AE) which can interact with a oneM2M server (CSE). Various open source and proprietary CSEs are available.

The OS-IoT package contains C++ functions and definitions which support the sending and receipt of oneM2M compatible messages. These functions support an API which allows applications to interact with the CSE at the level of oneM2M resource objects. This package can be used as a library to build IoT applications. The package also contains a demo command line application 'osiotcmd' which can be used to test CSEs and as a reference to see how to use the library functions.

OS-IoT is implemented using C++11 and is intended to be portable to a wide variety of platforms. During development, the package was tested on a PC/x64 running Ubuntu 16.04 LTS and a Dragonboard DB410c (ARM) running Linaro Debian Stretch.

1.1. Obtaining and Building OS-IoT

The OS-IoT package is available on Git. See the 'download', 'Build and Install' and 'Testing with a CSE' pages at www.os-iot.org

1.2. Dependencies

- CodeSynthesis XSD tree
(To build from the XSD files requires the full installation of CodeSynthesis XSD tree. If you just want to build from the C++ source files in the package then only the headers from CodeSynthesis XSD are needed).
- Apache XERCES
- Libcurl installed in default include path and default lib path for gcc
- OpenSSL in a version compatible with the linked Libcurl
- Jsoncpp - Amalgamated source files are included in this distribution. This does not need to be installed separately.
- Eidheim Simple Web Server - The header library is included in this distribution. This does not need to be installed separately.
- ASIO stand-alone library - The header library is included in this distribution. This does not need to be installed separately.

1.3. License

See the README file in the package for licensing and contribution information.

1.4. Reference for osiotcmd Application

The OS-IoT package contains an example command line application ('osiotcmd'). The source code demonstrates how to use the library and the application itself may be used for testing purposes.

The application takes options of two types:

- Command options that control the actions of the command, and
- Configuration options that configure settings for the command.

Note that because this application is only created for demo and testing purposes it performs very minimal validation on options. Options with invalid content will lead to undefined results.

See the examples below for some practical commands to perform common tasks.

1.4.1. Command Options

The options in this section act as commands to instruct the program what functions to perform. Commands may be chained in the same call by adding multiple options in order, e.g.:

```
$ ./osiotcmd -r -c
```

Note that there are two ways of specify the address of a resource. The older method (set using --a) specifies the <resourceAddress> and other parts of the address use hard-coded resourceNames. This method is convenient for some OS-IoT test scripts. The newer method (set using --A) specifies a <resourceAddress> in full (incorporating the cseRootAddress) and allows resourceNames to be freely chosen. This method is more generally useful. If the resourceAddress approach is used then the setting of the --a <resourceAddress> parameter is ignored and does not need to be correct.

Option	Name	Effect
--ae	ApplicationEntity	If <resourceAddress> is set then create an AE at <resourceAddress>. Use the <resourceName> if specified. If <resourceAddress> is not set then create an AE named "MY_SENSOR" at <cseRootAddress>
--container	Container	If <resourceAddress> is set then create a Container at <resourceAddress>. Use the <resourceName> if specified. If <resourceAddress> is not set then create an Container named "DESCRIPTOR" at <cseRootAddress>/MY_SENSOR
-c	Create	Creates: <ul style="list-style-type: none"> - An AE named 'MY_SENSOR' at <cseRootAddress> - A container named 'DESCRIPTOR' at <cseRootAddress>/MY_SENSOR - A contentInstance at <cseRootAddress>/MY_SENSOR/DESCRIPTOR
-d	Delete	Delete the resource at <resourceAddress> or <cseRootAddress>/MY_SENSOR if <resourceAddress> is not specified.
-u	Update	Update the AE resource at <resourceAddress> or <cseRootAddress>/MY_SENSOR if <resourceAddress> is not specified.
--uContainer	Update Container	Updates the Container resource at <resourceAddress> or <cseRootAddress>/MY_SENSOR/DESCRIPTOR if <resourceAddress> is not specified.

Option	Name	Effect
-s	Subscribe	<p>If <resourceAddress> is set then create a Subscription at <resourceAddress>. Use the <resourceName> if specified.</p> <p>If <resourceAddress> is not set then create a subscription named 'MY_SUBSCRIPTION'/in- at <cseRootAddress>/MY_SENSOR/DESCRIPTOR</p> <p>If <subscriptionPoa> is specified this will be included in the POA of the Subscription. This can set either local (e.g. <CSERootAddress>/<AE ResourceName>) addresses or global URIs. If no <poaAddr> is specified will use <cseRootAddress >/MY_SENSOR as the POA..</p>
--su	Subscribe and make URL out of <poaAddr> and <poaPort>	<p>As above, but includes a URL directly in the subscription resource which is constructed from the <poaAddr> and <poaPort> in the form: http://<poaAddr>:<poaPort>/</p>
--uSubscription	Update Subscription	Updates the Subscription resource at <resourceAddress> or <cseRootAddress>/MY_SENSOR/DESCRIPTOR/MY_SUBSCRIPTION if <resourceAddress> is not specified.
-p	PollingChannel	<p>If <resourceAddress> is set then create a PollingChannel at <resourceAddress>. Use the <resourceName> if specified.</p> <p>If <resourceAddress> is not set then create a PollingChannel named 'MY_POLLING_CHANNEL' at <cseRootAddress>/MY_SENSOR</p>
-uPollingChannel		Updates the PollingChannel resource at <resourceAddress> or <cseRootAddress>/MY_SENSOR/MY_POLLING_CHANNEL if <resourceAddress> is not specified.
--acp	AccessControlPolicy	<p>If <resourceAddress> is set then create an ACP at <resourceAddress>. Use the <resourceName> if specified.</p> <p>If <resourceAddress> is not set then create Create an AccessControlPolicy named 'MY_ACP' at <cseRootAddress>/MY_SENSOR</p>
--uAcp	Update ACP	Updates the ACP resource at <resourceAddress> or <cseRootAddress>/MY_SENSOR/MY_ACP if <resourceAddress> is not specified.
-r	Retrieve	Retrieve the resource at <resourceAddress>, if specified, or <cseRootAddress> if <resourceAddress> is not specified.
--ci	ContentInstance	<p>If <resourceAddress> is set then create a CI at <resourceAddress>.</p> <p>If <resourceAddress> is not set then create a CI at <cseRootAddress>/MY_SENSOR/DESCRIPTOR</p>

Option	Name	Effect
-l	Listen on polling channel	<p>If <resourceAddress> is set then poll a PollingChannel at <resourceAddress>.</p> <p>If <resourceAddress> is not set than poll a polling channel at: <cseRootAddress>/MY_SENSOR/ MY_POLLING_CHANNEL/pcu</p> <p>NOTE: Currently this is a blocking, never-ending polling option which will require the operation to be force-quit, e.g. with ^c or 'kill -9'</p> <p>NOTE: In the current version, the response from the polling channel is displayed if debug is enabled, but not decoded.</p>
-L	Listen on HTTP Server	<p>Will setup an HTTP server listening on port 18888, on any URL path, for notification requests. This server will be in a new thread from the main program. If the main program terminates the server thread will also terminate. Use with '-W' to force the server to stay open.</p> <p>e.g. \$./osiotcmd -L -W</p> <p>will create a server to listen and then wait for server activity.</p>
-W	Wait	<p>Stops further execution of the program except for the HTTP server, if it is running. Any commands after -W are not processed.</p> <p>NOTE: This will create a never-ending wait-loop which will require the operation to be force-quit, e.g. with ^c or 'kill -9'</p>

1.4.2. Configuration Options

The following options can change the configuration of the program. Where an option takes a value it should follow the option, e.g.

```
$ ./osiotcmd -h 127.0.0.1:9999 -x
```

Option	Effect	Default Value
-h	Host name – sets the host name and port to use	127.0.0.1:8080
-a	Address – sets the <cseRootAddress> This option may be used if osiotcmd's default resourceNames are applied. If you wish to use resourceNames other than the default then override this option using "-A" and, if you are creating a resource, "--rn".	/in-cse/in-name
--A	Address – set the <resourceAddress> Where this parameter is provided it overrides the <cseRootAddress> which will be ignored. If more than one "--A" parameter is provided the last one replaces all previous parameters, unless it starts with a "?". If the parameter starts with a "?" it is appended to the previous address to form a discovery string.	None
-f	From – sets the 'From' parameter	admin:admin
-x	Encode as XML	
-j	Encode as JSON	Default protocol is JSON
-q	Quiet: suppress debug output	Debug output is on
--rn	ResourceName – set the <resourceName> used for Create operations	None
--content	Content – set the content of a contentInstance. Only used with the --ci option.	Random string of format "Temp: xx" where xx is 00-99.
--https	Use certificate based HTTPS Note that security parameters should be specified if this option is used.	Off (use HTTP)
--httpsPsk	Use PSK based HTTPS As a side effect, this option will set the cipherList to "PSK". Note that security parameters should be specified if this option is used.	Off (use HTTP)

--noVerifyPeer	Use in conjunction with --https. Disable verification of the server's certificate	Peer verify On
--noVerifyHost	Use in conjunction with --https. Disable verification that the server's identify in its certificate matches the address used to contact it.	Host verify On
--caPath	Use in conjunction with --https. Specify path to the CA directory	None
--caInfo	Use in conjunction with --https. Specify CA file	None
--sslCert	Use in conjunction with --https. Specify file to use as client certificate	None
--sslKey	Use in conjunction with --https. Specify file containing the key for the client certificate.	None
--sslPasswd	Use in conjunction with --https. Specify password used to decode the file specified in --sslKey	None
--subjectANA	Use in conjunction with --https. Specify the allowed value for the server's subjectAltName	None
--cipherList	Use in conjunction with --https and --httpsPsk Specify the list of allowed ciphers (in openssl format). For example, to use the oneM2M TS0003 specified certificate cipher set to "ECDHE-ECDSA-AES128-SHA256". The ciphers specified should be compatible with the transport mode specified (certificate based Vs PSK security). If PSK security is to be used this must be set to "PSK" or a list of PSK ciphers.	Default openssl ciphers or "PSK" if --httpsPsk was previously specified.

	If <code>--httpsPsk</code> option is used this will overwrite any previously set value of the <code>cipherList</code> to "PSK".	
<code>--pskIdentity</code>	Use in conjunction with <code>--httpsPsk</code> Specify the PSK identity	None
<code>--pskKey</code>	Use in conjunction with <code>--httpsPsk</code> Specify the PSK key as a hex string. To use an ASCII string use <code>--pskKeyAscii</code>	None
<code>--pskKeyAscii</code>	Use in conjunction with <code>--httpsPsk</code> Specify the PSK key as an ASCII string. To use a hex string use <code>--pskKey</code>	None
<code>--passResCodes</code>	Comma separated list of OS-IoT Result codes that are "pass" results during a test. If the results of all operations are contained in this list then the command will return 0, else it will return non-zero Primarily useful as part of automated test scripts.	200,201,202
<code>--printTestRes</code>	Include to print "### STATUS CODE TEST PASS ###" or "=== STATUS CODE TEST FAIL ===" based on match of <code>passResCode</code> on termination.	Off
<code>--poaPort</code>	Set the port number to use for the HTTP notification server. Used for <code>-c</code> , <code>--su</code> and <code>-L</code> commands.	18888
<code>--poaAddr</code>	Sets the host name or IP address to indicate to the CSE as the Point of Access to send notifications to the client. Used for <code>-c</code> , <code>--su</code> and <code>--ae</code> commands.	Localhost
<code>--subscriptionPoa</code>	Specify the address to be used for creating a subscription using <code>"-s"</code> This can set either local (e.g. <code><CSERootAddress>/<AE ResourceName></code>) addresses or global URIs.	<code><cseRootAddress>/MY_SENSOR</code>
<code>--notRequest Reachable</code>	When used with <code>"-c"</code> to create an AE specifies that the created AE should be marked with <code>requestReachable=false</code> and no point of access (POA) is provided for the AE	Off
<code>--acrOriginator</code>	Originator (as URI string) used in the <code>AccessControlRule</code> created by <code>--acp</code>	*

<code>--acrOriginatorSelf</code>	Sets the <code>acrOriginator</code> to the value of "From"	
<code>--acrOperations</code>	Decimal value of the allowed operation code used in the <code>AccessControlRule</code> created by <code>--acp</code>	63 (= <code>accessControlAll</code>)
<code>--withAcp</code>	Specify the ACP to be assigned to a newly created resource. Currently only used for <code>-c</code> to create a Container	None

1.4.3. Command Examples

```
$ ./osiotcmd -h 127.0.0.1:9999 -f CAE01 -A CSE01ROOT --ae --rn NEW_AE
```

Create an AE on host 127.0.0.1, port 9999 at the address "CSE01ROOT" with resourceName "NEW_AE". Use the AE-ID of "CAE01".

```
$ ./osiotcmd -h 127.0.0.1:9999 -f CAE01 -A CSE01ROOT -r
```

Retrieve the resource at address "CSE01ROOT" on host 127.0.0.1, port 9999. Use the AE-ID of "CAE01".

```
$ ./osiotcmd -h 127.0.0.1:9999 -f CAE01 -A CSE01ROOT/NEW_AE -d
```

Delete the resource at address "CSE01ROOT/NEW_AE" on host 127.0.0.1, port 9999. Use the AE-ID of "CAE01".

```
$ ./osiotcmd -h 127.0.0.1:9999 -f CAE01 -A CSE01ROOT -r --httpsPsk --pskKeyAscii 123456 --pskIdentity test@onem2m.com
```

Retrieve the resource at address "CSE01ROOT" on host 127.0.0.1, port 9999. Use the AE-ID of "CAE01". Use symmetric PSK security with key (Kpsa) of "123456" and identity (Kpsaid) of "test@onem2m.com".

1.5. Contents of the Repository

The root directory of the repository contains the code and build commands for the library and the `osiotcmd` shell command. This uses information from the "cdt" directory which includes files and tools related to translating the oneM2M XSD protocol definitions in to C++.

The directory "functionaltests" contains scripts for testing the library against locally running CSEs (these are not the TS0013 tests). See the readme in that directory.

The directory "ts0013tests" contains scripts to test according to TS-0013. See the readme in that directory.

The directory "demoweb1ae" contains a demo (using one AE for all roles) which can show OS-IoT operation over the public Internet.

Various dependencies which are packaged with the distribution are included in other directories.

2. Technical Feature Overview

2.1. Supported Features

OS-IoT supports oneM2M compliant AE implementations including the following features:

- HTTP network transport
- PSK or Certificate based HTTPS transport when acting as an HTTP client
- XML or JSON serialization
- AE, CSE-Base, Container, Content Instance, Subscription, Polling Channel and Access Control Policy resources
- Create/Update/Retrieve/Delete operations for resources on a remote CSE for the supported resource types
- Request Reachable support for subscriptions including use of an HTTP server on the AE

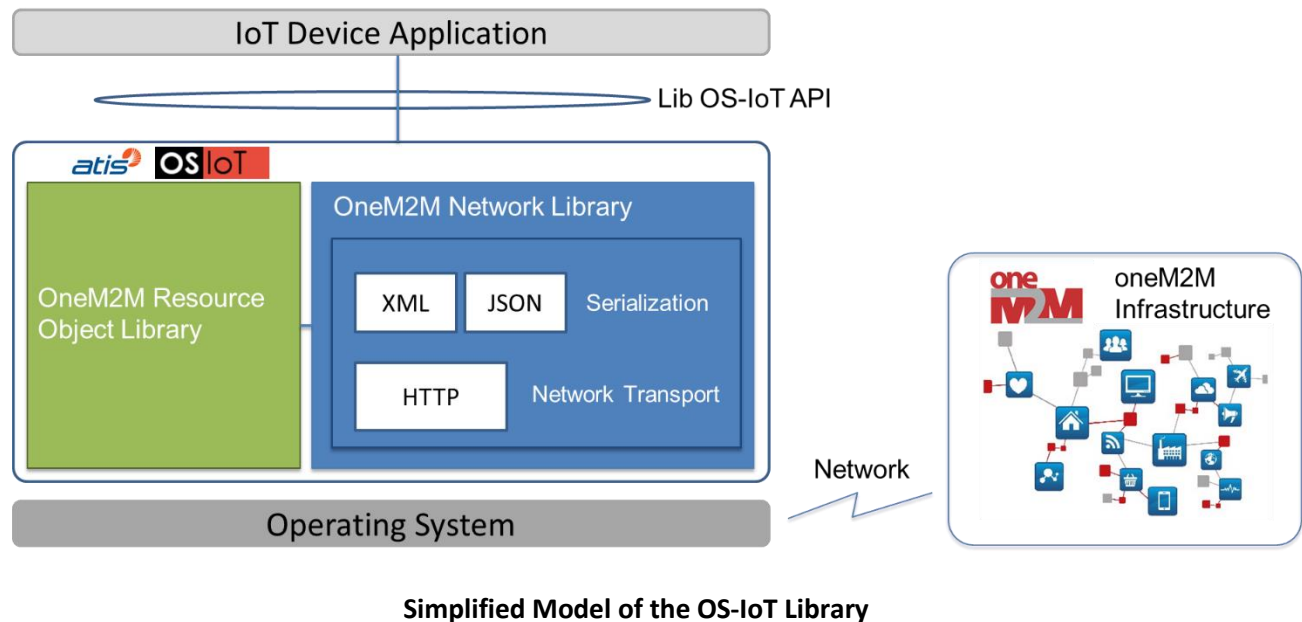
For details of the features supported please see subsequent sections.

2.2. Conceptual Model

The figure below shows a high-level view of the OS-IoT library. The library is intended to run on an IoT device and provide the embedded application with a simple method of reading and writing resources on a oneM2M compliant CSE.

The library contains two main aspects:

- A resource object library of oneM2M compliant resources instantiated as C++ objects
- Network communication procedures to allow these resources to be written to and read from a CSE.



A developer who wishes to program a oneM2M compatible client device can link the library as part of their application. The application can then perform actions such as creating a local oneM2M resource to represent the data that it wishes to write to the network and then using the network library to send this to a local CSE. For details on the library API please see subsequent sections.

3. API Specification

The formal specification of the programming API for the library is available at:

<http://os-iot.org/apireference/>

3.1. Network Function API

The API for network functions consists of functions defined in the onem2m namespace. See “Library Usage” at the above link or the example applications to see how to use them.

3.2. Resource Object Library API

The resource object library API is a set of C++ headers and class implementations automatically generated from oneM2M standard XSD files using Codesynthesis XSD Tree. The API specification available at the link above defines the usage. Further information which may be useful may be obtained from:

- The C++ header files contained in the project’s ‘cdt’ directory
- The modified oneM2M XSD files contained in the same project directory
- The Codesynthesis XSD Tree documentation, in particular the “C++/Tree Mapping User Manual”

For the key oneM2M objects and parameters inspection of the example application included in the package should illustrate how to create and read the most important fields from objects.

Note that the object naming in the API follows the “long names” defined by oneM2M. The long names are automatically mapped to and from the corresponding short names as part of the serialization functions of the library.

3.2.1. General Guidance

When manipulating oneM2M resource objects please keep the following points in mind:

- The OS-IoT library does not enforce presence of mandatory parameters in objects or ensure that objects comply to restrictions like ranges or naming conventions specified by oneM2M.
- When an object is to be sent, it is the responsibility of the application to ensure that the object is populated with all parameters that are mandatory, according to the onem2m specifications, in that context. For example, when creating an AE object, the application should ensure that the “requestReachable” parameter is set appropriately.
- When objects are received the OS-IoT library will map all supported parameters that it can interpret according to the correct type to the resource object passed to the application. All parameters (and sub parameters) in this object are treated as optional, regardless of their status in the oneM2M standard. Applications should check received objects according to their particular requirements. Note that the objects in the API encode attributes differently depending on their optionality and multiplicity. This is explained in the API specification. Make sure you use the appropriate access method for the parameter and check for presence of optional parameters before attempting to read them. Failure to do this will cause memory errors if parameters are not present.

3.3. Threading

Except for the HTTP server (see below), all OS-IoT procedures run in the same thread as the calling code. All procedures are blocking until a result is returned or they time-out. The OS-IoT library does not support multithreaded operation.

The HTTP server used to support listening for notifications runs in its own thread which is created and managed by the OS-IoT library.

3.4. Error and Exception Handling

OS-IoT attempts to avoid generating errors based on received data by interpreting as much data as it can understand and silently ignoring data that is not supported or not compliant to oneM2M standards. In normal operation, OS-IoT should not throw exceptions.

In error situations OS-IoT does not attempt to perform error recovery itself. Exceptions thrown by the library or by its dependencies are generally not handled in the library and will be passed back to the calling application. Applications should define their own error processing procedures.

A different exception handling behaviour is applied for the HTTP server thread. Exceptions that occur during the establishment of the HTTP server are reported in the result of the `startHttpServer()` function. Exceptions that occur after this are caught (and should be logged to `std::out`) by the OS-IoT library and execution of the server thread stopped without the exception being rethrown. The calling application can check the status of the HTTP server thread using the `getHttpServerRunning()` function.

Correct handling of errors, particularly those caused by receipt of malicious data, is important to protect the security of networked IoT devices. Application designers are recommended to give careful attention to the design and testing of this aspect.

4. Protocol Compliance Statement

This section shows the level of support for oneM2M protocol elements in OS-IoT. It first shows the general parameters of the Request Primitive and Response Primitive. It then shows the support for protocol fields in the contents of the Request Primitive and Response Primitive.

4.1. Request Primitive

Request operations are sent by OS-IoT. The following table shows the status of parameters.

Primitive Parameter	Multiplicity	Support in HTTP Transport	Note
Operation	1	Supported	
To	1	Supported	Treated as oneM2M address
From	0..1	Supported	
Request Identifier	1	Supported	
Resource Type	0..1	Supported	Must match resource contained in Content
Content	0..1	Supported	See section 3 for details
Role IDs	0..1	Not supported	
Originating Timestamp	0..1	Not supported	
Request Expiration Timestamp	0..1	Not supported	
Result Expiration Timestamp	0..1	Not supported	
Operation Execution Time	0..1	Not supported	
Response Type	0..1	Not supported	
Result Persistence	0..1	Not supported	
Result Content	0..1	Not supported	
Event Category	0..1	Not supported	
Delivery Aggregation	0..1	Not supported	
Group Request Identifier	0..1	Not supported	

Primitive Parameter	Multiplicity	Support in HTTP Transport	Note
Filter Criteria	0..1	Supported as a query string in the "To" address	
Discovery Result Type	0..1	Not supported	
Tokens	0..1	Not supported	
Token IDs	0..1	Not supported	
LocalTokenIDs	0..1	Not supported	
Token Request Indicator	0..1	Not supported	
Group Request Target Members	0..1	Not supported	
Authorization Signature Indicator	0..1	Not supported	
Authorization Signatures	0..1	Not supported	
Authorization Relationship Indicator	0..1	Not supported	

4.2. Primitive Response

Primitive responses are received by OS-IoT. The following table shows the status of support.

Primitive Parameter	Multiplicity	Support in HTTP Transport	Note
Response Status Code	1	Supported	Response code is handled at the level of the HTTP response code. The X-M2M-RSC header is not supported.
Request Identifier	1	Ignored	

Primitive Parameter	Multiplicity	Support in HTTP Transport	Note
Content	0..1	Supported	See section 4 for details
To	0..1	Ignored	
From	0..1	Ignored	
Originating Timestamp	0..1	Ignored	
Result Expiration Timestamp	0..1	Ignored	
Event Category	0..1	Ignored	
Content Status	0..1	Ignored	
Content Offset	0..1	Ignored	
Assigned Token Identifiers	0..1	Ignored	
Token Request Information	0..1	Ignored	
Authorization Signature Request Information	0..1	Ignored	

4.3. Resources in Request Primitive Content

The following resources are supported by OS-IoT when sending a Request Primitive.

“Not tested” means that it is possible that the correct serialization may be generated but that this hasn’t been tested and support should not be assumed.

4.3.1. Universal/Common Attributes

Attribute Name	Request Optionality		Support
	Create	Update	
@resourceName	O	NP	Supported
<i>resourceType</i>	NP	NP	
<i>resourceID</i>	NP	NP	
<i>parentID</i>	NP	NP	

Attribute Name	Request Optionality		Support
	Create	Update	
<i>accessControlPolicyIDs</i>	O	O	Supported
<i>creationTime</i>	NP	NP	
<i>expirationTime</i>	O	O	Not tested
<i>lastModifiedTime</i>	NP	NP	
<i>labels</i>	O	O	Supported
<i>announceTo</i>	O	O	Not tested
<i>announcedAttribute</i>	O	O	Not tested
<i>dynamicAuthorizationConsultationIDs</i>	O	O	Not tested

4.3.2. AE

Resource Specific attributes

Attribute Name	Request Optionality		Support
	Create	Update	
<i>appName</i>	O	O	Not tested
<i>App-ID</i>	M	NP	Supported
<i>AE-ID</i>	NP	NP	
<i>pointOfAccess</i>	O	O	Supported
<i>ontologyRef</i>	O	O	Not tested
<i>nodeLink</i>	O	O	Not tested
<i>requestReachability</i>	M	O	Supported
<i>contentSerialization</i>	O	O	Supported (Note 1)
<i>e2eSecInfo</i>	O	O	Not tested

Note 1: The contentSerialization parameter is supported by OS-IoT. However, support for this parameter seems to be inconsistent on servers. IoTDM appears to reject values specified by current versions of the oneM2M specifications. Eclipse OM2M seems to ignore the parameter.

4.3.3. Container

Resource Specific attributes

Attribute Name	Request Optionality		Support
	Create	Update	
<i>maxNrOfInstances</i>	O	O	Not tested
<i>maxByteSize</i>	O	O	Not tested
<i>maxInstanceAge</i>	O	O	Not tested
<i>currentNrOfInstances</i>	NP	NP	
<i>currentByteSize</i>	NP	NP	
locationID	O	O	Not tested
<i>ontologyRef</i>	O	O	Not tested
<i>disableRetrieval</i>	O	NP	Not tested

4.3.4. Content Instance

Resource Specific attributes

Attribute Name	Request Optionality	Support
	Create	
<i>contentInfo</i>	O	Supported
<i>contentSize</i>	NP	
<i>contentRef</i>	O	Not tested
<i>ontologyRef</i>	O	Not tested
<i>content</i>	M	Supported (treated as type xs:string)

4.3.4. Subscription

Resource Specific Attributes

Attribute Name	Request Optionality		Support
	Create	Update	
<i>eventNotificationCriteria</i>	O	O	Supported (Note 1)
<i>expirationCounter</i>	O	O	Supported
<i>notificationURI</i>	M	O	Supported
<i>groupID</i>	O	O	Not tested
<i>notificationForwardingURI</i>	O	O	Not tested
<i>batchNotify</i>	O	O	Not supported
<i>rateLimit</i>	O	O	Not supported
<i>preSubscriptionNotify</i>	O	NP	Not supported
<i>pendingNotification</i>	O	O	Not supported
<i>notificationStoragePriority</i>	O	O	Not supported
<i>latestNotify</i>	O	O	Not tested
<i>notificationContentType</i>	O	O	Not supported
<i>notificationEventCat</i>	O	O	Not supported
<i>subscriberURI</i>	O	NP	Not supported

NOTE 1: For the eventNotificationCriteria only the 'notificationEventType' parameter is supported.

4.3.5. Polling Channel

Note: Polling Channel is not supported by IoTDM at the time of writing.

4.3.6. Access Control Policy

The following tables indicate the supported attributes.

Attribute Name	Support
Privileges	Supported
selfPrivileges	Supported

The following table shows the support of the object m2m:accessControlRule.

Attribute Name	Support
accessControlOriginators	Supported
accessControlOperations	Supported
accessControlContexts	Supported (Note 1)
accessControlContexts/accessControlWindow	Supported
accessControlContexts/accessControlIpAddresses	Supported
accessControlContexts/accessControlIpAddresses/ipv4Addresses	Supported
accessControlContexts/accessControlIpAddresses/ipv6Addresses	Supported
accessControlContexts/accessControlLocationRegions	Supported (Note 2)
accessControlAuthenticationFlag	Supported (Note 1)
accessControlObjectDetails	Not supported
accessControlObjectDetails/resourceType	Not supported
accessControlObjectDetails/specializationID	Not supported
accessControlObjectDetails/childResourceType	Not supported

Note 1: Support for these attributes is inconsistent in CSEs. For maximum compatibility with different CSEs it is recommended that they are not populated in request messages.

Note 2: Only supported at the level of a list of countries.

4.4. Resources in Response Primitive Content

The following resources are supported by OS-IoT when receiving a Response Primitive.

“Not tested” means that support is included in the code, but the correct mapping has not, yet, been verified.

4.4.1. Universal/Common Attributes

Attribute Name	Support
@resourceName	Supported
<i>resourceType</i>	Supported
<i>resourceID</i>	Supported
<i>parentID</i>	Supported
<i>accessControlPolicyIDs</i>	Supported

Attribute Name	Support
<i>creationTime</i>	Supported
<i>expirationTime</i>	Supported
<i>lastModifiedTime</i>	Supported
<i>labels</i>	Supported
<i>announceTo</i>	Not tested
<i>announcedAttribute</i>	Not tested
<i>dynamicAuthorizationConsultationIDs</i>	Not tested

4.4.2. AE

Resource Specific attributes

Attribute Name	Support
<i>appName</i>	Not tested
<i>App-ID</i>	Supported
<i>AE-ID</i>	Supported
<i>pointOfAccess</i>	Not tested
<i>ontologyRef</i>	Not tested
<i>nodeLink</i>	Not tested
<i>requestReachability</i>	Not tested
<i>contentSerialization</i>	Not tested
<i>e2eSecInfo</i>	Not tested

4.4.3. Container

Resource Specific attributes

Attribute Name	Support
----------------	---------

<i>maxNrOfInstances</i>	Not tested
<i>maxByteSize</i>	Not tested
<i>maxInstanceAge</i>	Not tested
<i>currentNrOfInstances</i>	Not tested
<i>currentByteSize</i>	Not tested
locationID	Not tested
<i>ontologyRef</i>	Not tested
<i>disableRetrieval</i>	Not tested
<i>stateTag</i>	Supported

4.4.4. Content Instance

Resource Specific attributes

Attribute Name	Support
<i>contentInfo</i>	Supported
<i>contentSize</i>	Supported
<i>contentRef</i>	Not tested
<i>ontologyRef</i>	Not tested
<i>content</i>	Supported (treated as type xs:string)
<i>stateTag</i>	Not tested

4.4.5. CSE Base

Data Type	Attribute Name	Support
m2m:cseTypeID	<i>cseType</i>	Not tested
m2m:ID	<i>CSE-ID</i>	Not tested
list of m2m:resourceType	<i>supportedResourceType</i>	Supported
m2m:poaList	<i>pointOfAccess</i>	Not tested
xs:anyURI	<i>nodeLink</i>	Not tested
list of xs:anyURI	<i>dynamicAuthorizationConsultationIDs</i>	Not tested
m2m:e2eSecInfo	<i>e2eSecInfo</i>	Not tested

4.4.6. Subscription

Resource Specific Attributes

Attribute Name	Support
<i>eventNotificationCriteria</i>	Not tested
<i>expirationCounter</i>	Not tested
<i>notificationURI</i>	Not tested
<i>groupID</i>	Not tested
<i>notificationForwardingURI</i>	Not tested
<i>batchNotify</i>	Not supported
<i>rateLimit</i>	Not supported
<i>preSubscriptionNotify</i>	Not supported
<i>pendingNotification</i>	Not supported
<i>notificationStoragePriority</i>	Not supported
<i>latestNotify</i>	Not tested

<i>notificationContentType</i>	Not supported
<i>notificationEventCat</i>	Not supported
<i>subscriberURI</i>	Not supported

4.4.7. Polling Channel

No Resource Specific Attributes

4.4.8. AccessControlPolicy

The following tables indicate the supported attributes.

Attribute Name	Support
Privileges	Supported
selfPrivileges	Supported

The following table shows the support of the object m2m:accessControlRule.

Attribute Name	Support
accessControlOriginators	Supported
accessControlOperations	Supported
accessControlContexts	Not tested
accessControlContexts/accessControlWindow	Not tested
accessControlContexts/accessControlIpAddresses	Not tested
accessControlContexts/accessControlIpAddresses/ipv4Addresses	Not tested
accessControlContexts/accessControlIpAddresses/ipv6Addresses	Not tested
accessControlContexts/accessControlLocationRegions	Not tested
accessControlAuthenticationFlag	Not tested
accessControlObjectDetails	Not supported
accessControlObjectDetails/resourceType	Not supported
accessControlObjectDetails/specializationID	Not supported
accessControlObjectDetails/childResourceType	Not supported

4.5. Notification

The following table shows parameter support for receipt of notification primitives.

Aggregated notifications are not supported.

Parameter	Support
notificationEvent	Supported

representation	Supported for object type contentInstance only
operationMonitor	Not tested
notificationEventType	Supported
verificationRequest	Supported
subscriptionDeletion	Supported
subscriptionReference	Supported
creator	Not tested
notificationForwardingURI	Not tested
IPEDDiscoveryResult	Not supported

4.6. [listOfURIs](#)

This listOfURIs object can be sent in the response to a discovery request. This is supported for both XML and JSON.

Appendix A – Additional References

Architecture Presentation

An overview presentation of the library architecture is available here:

<http://access.atis.org/apps/org/workgroup/os-iot/download.php/35503/latest>

Appendix B – Compatibility Notes with CSE Implementations

B.1. Eclipse OM2M

The following areas limit compatibility with OS-IoT:

- In the notification primitive, OM2M sends a parameter tagged “rss” as part of the notificationEvent (“nev”) structure. This tag is not defined in oneM2M. It may be an alternative encoding of the notificationEventType (“net”).
- If the parameter org.eclipse.om2m.notification.mmt is set to “application/json” to configure OM2M to send notifications encoded in JSON format (see OM2M bug ID 507443) then OM2M will encode the notification in JSON, but may not use the correct value of the “Content-Type” header. In testing, the verification event was sent with the correct Content-Type, but a notification for due to a subscription was sent with Content-Type = application/xml. To work around this, keep the default (XML) setting for org.eclipse.om2m.notification.mmt.
- The ACOR tag in ACR is incorrectly encoded in XML responses (should be list of space separated list of values under one tag, but is in fact listed as multiple tags). OS-IoT will ignore this malformed XML. See OM2M bug ID 531929.
- PollingChannel functionality is not supported. Though a pollingChannel resource can be created, an attempt to send a GET operation to the pollingChannelURI creates an error.

Further areas to note are:

- The default ‘admin:admin’ logon in the From Parameter does not conform to the requirements of TS0001. OS-IoT does not enforce conformance to TS0001 and permits the use of this parameter.
- The parameter “contentSerialization” in the AE resource seems to be ignored – rather the serialization for notifications is based on a local configuration (see OM2M bug ID 507443).
- When creating a pollingChannel resource an ACPI is added to the resource even though this is not defined as part of the pollingChannel resource in the oneM2M specifications (see TS0001 s9.6.21). Also, a parameter tagged ‘pcu’ (pollingChannelURI perhaps) is added to the response. This parameter is not defined in oneM2M specifications. OS-IoT ignores these additional parameters.

B.2. IoTDM

The following areas limit compatibility with OS-IoT:

- PollingChannel resource type is not supported
- Notifications are sent without the notification (“sgn”) JSON tag which should be included. This tag must be inferred by the receiver. The Content-Type header is “application/vnd.onem2m-ntfy+json”. **NOTE:** OS-IoT includes a work-around for this case – if the first attempt to decode

the notification according to oneM2M specifications fails, and if the Content-Type matches the value used by IoTDM then the missing tag will be inserted and a second attempt to decode the notification will be performed.

- The parameter contentSerialization in the AE resource does not support the values specified by oneM2M. The values supported are the strings “json” and “xml” (case insensitive). See IoTDM bug 9002. By default IoTDM will use JSON.

B.3. Ocean IoT Mobius V2.0.0

- Notifications to the AE PoA in the AE resource are not supported. The AE must specify a URL in the individual Subscription resource.
- In some versions the notification operation contains encoding mistakes that impact the <representation> and <notificationEventType> fields meaning that it can't be decoded by OS-IoT. These are planned for correction in later versions of Mobius

B.4 Interdigital oneMPOWER

- Some versions only accept result code 201 (one2mHttpCREATED) and not 200 (onem2mHttpOK) as a successful acknowledgement of a notification validation operation.
- AccessControlContexts (acco) is only supported if sent as a single object in JSON. The XSD defines this as having multiplicity therefore the correct JSON encoding (used by OS-IoT) is an array containing one or more objects. Do not populate the AccessControlContext if sending an AccessControlPolicy to oneMPOWER. Even for an acco encoded how oneMPOWER expects, populating an accessControlLocationRegion or an accessControlAuthenticationFlag in an accessControlPolicy can generate an error.
- In a JSON encoded notification request the “representation” (rep) element may be sent as a string containing a JSON object rather than as a JSON object directly. The OS-IoT library accepts both encodings.

Appendix C – Resource Support Reported by Tested CSEs

Performing a GET on the CSE generates a list of supported resource types. The following table shows the resource types self-reported as supported by the CSEs listed.

ID	Resource Name	OM2M V1.2.0	IoTDM Carbon	One MPower	Mobius V2.0.0 Commit 35405b1	Open MTC
1	accessControlPolicy	Yes	Yes	Yes	Yes	Yes
2	AE	Yes	Yes	Yes	Yes	Yes
3	container	Yes	Yes	Yes	Yes	Yes
4	contentInstance	Yes	Yes	Yes	Yes	Yes
5	CSEBase	Yes	Yes	Yes		Yes
6	delivery					
7	eventConfig			Yes		
8	execInstance					
9	group	Yes	Yes	Yes	Yes	
10	locationPolicy				Yes	
11	m2mServiceSubscriptionProfile			Yes		
12	mgmtCmd			Yes		
13	mgmtObj			Yes		
14	node	Yes	Yes	Yes		
15	pollingChannel	Yes		Yes		
16	remoteCSE	Yes		Yes	Yes	Yes
17	request	Yes		Yes	Yes	
18	schedule			Yes		
19	serviceSubscribedAppRule			Yes		
20	serviceSubscribedNode			Yes		
21	statsCollect			Yes		
22	statsConfig					
23	subscription	Yes	Yes	Yes	Yes	Yes

ID	Resource Name	OM2M V1.2.0	IoTDM Carbon	One MPower	Mobius V2.0.0 Commit 35405b1	Open MTC
24	semanticDescriptor			Yes	Yes	Yes
25	notificationTargetMgmtPolicyRef					
26	notificationTargetPolicy					
27	policyDeletionRules					
28	flexContainer	Yes (not in V1.0.0)		Yes		
29	timeSeries				Yes	
30	timeSeriesInstance				Yes	
31	role					
32	token					
33	trafficPattern					
34	dynamicAuthorizationConsultation					
10001	accessControlPolicyAnnc			Yes		
10002	AEAnnc			Yes		
10003	containerAnnc			Yes		
10004	contentInstanceAnnc			Yes		
10009	groupAnnc			Yes		
10010	locationPolicyAnnc					
10013	mgmtObjAnnc			Yes		
10014	nodeAnnc			Yes		
10016	remoteCSEAnnc			Yes		
10018	scheduleAnnc			Yes		
10024	semanticDescriptorAnnc					
10028	flexContainerAnnc					
10029	timeSeriesAnnc					

ID	Resource Name	OM2M V1.2.0	IoTDM Carbon	One MPower	Mobius V2.0.0 Commit 35405b1	Open MTC
10030	timeSeriesInstanceAnnc					
10033	trafficPatternAnnc					
10034	dynamicAuthorizationConsultationAnnc					
	Additional (proprietary?)			2100		