API calls Examples

If you have any question, feel free to ask us at https://iotdm.slack.com/signup.

This document provides examples of oneM2M CRUD operations using JSON over HTTP.

CoAP CRUDs examples will be provided soon.

In these examples we assume the existence of a CSE called InCSE1, active on port 8282. If you are playing with Cisco Devnet, the port number may be 8888, ask anyone in the iotdm team if you are not sure.

In the Lithium release (June 2015), we support only the following oneM2M resource types:

- Container
- ContentInstance
- AE
- CSE
- Subscription

For details about resource types and their attributes, see TS-0001 (http://www.onem2m.org/images/files/deliverables/TS-0001-Functional_Architecture-V1_6_1.pdf) sect 9.6

TS-0001 draft version contains more information than the published version. (http://www.onem2m.org/technical/latest-drafts)

Please remember these few rules when building a tree:

- A resource which doesn't have an existing parent can't be created
- Some resources can't be created under others. For example, a resource type ***contentInstance*** can't be created under a resource type ***AE*** (application entity).
- We use short names attributes as defined in oneM2M TS004 Sect 8.2 v1 (http://www.onem2m.org/technical/published-documents). This produces very short and compact messages suitable for resource constrained IoT devices. There is also a short name XSD file provided by OneM2M (http://member.onem2m.org/Application/documentApp/documentinfo/?documentId=13134&fromList=Y), you may need to register a free account before download it.

As you construct a request message, the information may go in 3 different places: the URL (including the query string), the HTTP header and the JSON payload. A list of supported fields is provided at the end of this document.

CRUD examples:

Create a resource *Application Entity (AE)* called "TestAE" under a *CSE* named "InCSE1"

Request

```
POST /InCSE1 HTTP/1.1
Host: X.X.X.X:8282
```

Headers:

```
Content-Type: application/vnd.onem2m-res+json;ty=2
X-M2M-Origin: //localhost:10000
X-M2M-RI: 12345
X-M2M-NM: TestAE
Cache-Control: no-cache
Postman-Token: b0e347e9-ccc8-c1b8-c566-eb1fb7a0e76f
```

JSON Body:

```
{"m2m:ae":{
    "api":"testAppId",
    "apn":"testAppName",
    "or":"http://ontology/ref",
    "rr":true
    }
}
```

Response

HTTP Header:

Content-Length → 120
Content-Location →/InCSE1/TestAE
Content-Type →application/vnd.onem2m-res+json;ty=2;charset=ISO-8859-1
Server → Jetty(8.1.15.v20140411)
X-M2M-RI → 12345
X-M2M-RSC → 2001

JSON Body:

```
{
 "m2m:ae": {
```

```
  "ct": "20151029T220112Z",
  "aei": "TestAE",
  "ri": "3",
  "lt": "20151029T220112Z",
  "pi": "/InCSE1/2",
  "api": "testAppId"
 }
}
```

Update "TestAE"'s attribute: "ontology reference(or)" to null
This will remove the "or" attribute of "TestAE"

Request

```
PUT /InCSE1/TestAE HTTP/1.1
Host: X.X.X.X:8282
Headers:
Content-Type: application/vnd.onem2m-res+json
X-M2M-Origin: //localhost:10000
X-M2M-RI: 12345
Cache-Control: no-cache
Postman-Token: 32c7611b-2992-e373-38d9-01d1fd72137b
```

JSON Body:

```
{"m2m:ae":{
    "or":null
    }
}
```

Response

HTTP Header:
Access-Control-Allow-Headers → accept
Access-Control-Allow-Methods → GET, PUT, POST, DELETE, HEAD
Access-Control-Allow-Origin → *
Content-Length → 171
Content-Type →application/vnd.onem2m-res+json;ty=2;charset=ISO-8859-1
Server → Jetty(8.1.15.v20140411)
X-M2M-RI → 12345

X-M2M-RSC → 2004

Json Body:


```
{
 "m2m:ae": {
  "rr": true,
  "ct": "20151030T001812Z",
  "aei": "TestAE",
  "ty": 2,
  "ri": "3",
  "lt": "20151030T041722Z",
  "pi": "/InCSE1/2",
  "api": "testAppId",
  "rn": "TestAE",
  "apn": "testAppName"
 }
}
```


Example of GET

```
GET /InCSE1/TestAE?rcn=4&drt=1 HTTP/1.1
Host: X.X.X.X:8282
```

Headers:

```
Content-Type: application/vnd.onem2m-res+json
X-M2M-Origin: //localhost:10000
X-M2M-RI: 12345
Cache-Control: no-cache
Postman-Token: 53edde3f-d028-9e54-395c-f91761a97125
```

Response

Headers:

Access-Control-Allow-Headers → accept
Access-Control-Allow-Methods → GET, PUT, POST, DELETE, HEAD
Access-Control-Allow-Origin → *
Content-Length → 317
Content-Type →application/vnd.onem2m-res+json;ty=2;charset=ISO-8859-1
Server → Jetty(8.1.15.v20140411)
X-M2M-RI → 12345
X-M2M-RSC → 2000

JSON Body:


```
{
  "m2m:ae": {
    "rr": true,
    "ct": "20151030T001812Z",
    "aei": "TestAE",
    "ch": [
      {
        "m2m:cnt": {
          "ct": "20151030T042058Z",
          "st": 0,
          "ty": 3,
          "cbs": 0,
          "ri": "4",
          "lt": "20151030T042058Z",
          "pi": "/InCSE1/3",
          "rn": "TestContainer",
          "cni": 0
        }
      }
    ],
    "ty": 2,
    "ri": "3",
    "lt": "20151030T042058Z",
    "pi": "/InCSE1/2",
    "api": "testAppId",
    "rn": "TestAE",
    "apn": "testAppName"
  }
}
```


Example of delete

This example deletes the resource /InCSE1/TestAE and everything below i
t [which means including the child Container in the example above]

```
DELETE /InCSE1/TestAE HTTP/1.1
Host: X.X.X.X:8282


Content-Type: application/vnd.onem2m-res+json

X-M2M-Origin: //localhost:10000
X-M2M-RI: 12345
Cache-Control: no-cache
Postman-Token: 2856fa62-6d34-693a-30f7-b115e27ef6db
```

Response

200 OK

HTTP Header:

Access-Control-Allow-Headers → accept
Access-Control-Allow-Methods → GET, PUT, POST, DELETE, HEAD
Access-Control-Allow-Origin → *
Content-Length → 171
Content-Type → application/vnd.onem2m-res+json;ty=2;charset=ISO-8859-1
Server → Jetty(8.1.15.v20140411)
X-M2M-RI → 12345
X-M2M-RSC → 2002

JSON Body:


```
{
  "m2m:ae": {
    "rr": true,
    "ct": "20151030T001812Z",
    "aei": "TestAE",
    "ty": 2,
    "ri": "3",
    "lt": "20151030T042058Z",
    "pi": "/InCSE1/2",
    "api": "testAppId",
    "rn": "TestAE",
    "apn": "testAppName"
  }
}
```


Resource subscription example

This example creates a subscription under /InCSE1/TestAE/myContainer1000 . If something changes in /InCSE1/TestAE/myContainer1000, two notifications will be sent to the notificationURI (nu) addresses.

```
POST /InCSE1/TestAE/myContainer1000 HTTP/1.1
Host: 64.103.37.47:8888
Content-Type: application/json
X-M2M-Origin: //localhost:10000
X-M2M-RI: 12345
X-M2M-NM: mySubscription
Cache-Control: no-cache

{"m2m:sub":{

    "nu":["http://Notification/destination1","http://Notification/desti
nation2"]

    }

}
```

Response

201 created

JSON Body:


```
{
  "m2m:sub": {
    "ct": "20151030T043243Z",
    "ri": "7",
    "lt": "20151030T043243Z",
    "pi": "/InCSE1/6"
  }
}
```

List of supported query fields

As mentioned above, when you construct a request message, the information may go in 3 different places: the URL (including the query string), the HTTP header and the JSON payload.

HTTP Headers:

See sect 6.4 of TS-009 (HTTP binding doc) to see how the headers are constructed.

Supported HTTP headers:

X-M2M-NM: used in a CREATE request to name the resource to be created
X-M2M-RSC: contains the response codes to queries
X-M2M-Origin : mapped to the ***From*** parameter of the request/response primitive.
X-M2M-RI: mapped to the ***Request Identifier*** parameter.
X-M2M-RTU: mapped to the *notificationURI* element of the ***Response Type*** parameter if applicable. If there are more than one value in the element, then the values shall be combined with "&" character.

Query string field: Primitive parameter short names (see TS-004 for details).

Supported parameters in the request query string:

Parameter Name	Short Name
Resource Type	ty
Response Type	rt
Result Content	rcn
createdBefore	crb
createdAfter	cra
modifiedSince	ms
unmodifiedSince	us
stateTagSmaller	sts
stateTagBigger	stb
labels	lbl
resourceType	rty
sizeAbove	sza
sizeBelow	szb
contentType	cty
limit	lim
attribute	atr
filterUsage	fu
Discovery Result Type	drt

Value of resource type (ty) in URL query string	Interpretation
2	AE
3	container
4	contentInstance
23	subscription

Discovery Result Type (drt) may be hierarchical (=1) or non-hierarchical (=2)

resultContent (rcn) expresses the type of content we want to receive in the response to a query

From TS-004

Value	Interpretation
0	nothing
1	attributes
2	hierarchical address
3	hierarchical address and attributes
4	attributes and child resources
5	attributes and child resource references
6	child resource references
7	original resource

JSON body short names:

The attributes supported in JSON follow the shortname convention as defined in oneM2M TS-004 Sect 8.2. A subset is listed here for Convenience. Please refer to the latest oneM2M specification for up to date list of short names

Attribute Name	Occurs in	Short Name
creationTime	All	***ct***
lastModifiedTime	All	***lt***
parentID	All	***pi***
resourceID	All	***ri***
stateTag	container, contentInstance, delivery, request	***st***
resourceName	All	***rn***
App-ID	AE	***api***
AE-ID	AE	***aei***
appName	AE	***apn***
ontologyRef	AE, container, contentInstance	***or***
nodeLink	AE, CSEBase, remoteCSE	***nl***
creator	container, contentInstance,eventConfig, group, pollingChannel, statsCollect, statsConfig, subscription	***cr***
maxNrOfInstances	container	***mni***
maxByteSize	container	***mbs***
maxInstanceAge	container	***mia***
currentNrOfInstances	container	***cni***

Attribute Name	Occurs in	Short Name
operation	request	***opn***
requestID	request	***rid***
notificationURI	subscription	***nu***
notificationForwardingURI	subscription	***nfu***
latestNotify	subscription	***ln***
notificationContentType	subscription	***nct***
notificationEventCat	subscription	***nec***
subscriberURI	subscription	***su***

Latest changes:

1. add resourceType at the beginning of each JSON payload
2. When create the resource, move the “ty=x” from the URI part into the “Content-Type”Header
3. Put the location of the new created resource into the Response Header “Content-Location”
4. Response’s payload only returns attributes “assigned by System”, so it will not return the attributes provided by the user in the Request payload.