Using the oneTRANSPORT Service Portal API

Introduction

The oneTRANSPORT Data Marketplace provides a means for organisations to publish dynamic (i.e. regularly changing or real-time) data and Static resource files (i.e. Asset location / Dataset models) for discovery and consumption by others. The primary means for publishing and consuming data through the platform is via open system APIs. The Portal is a web service that enables developers to engage with the oneTRANSPORT service via a web browser, both to interact with data directly and to access detailed documentation about the system APIs.

API Authentication

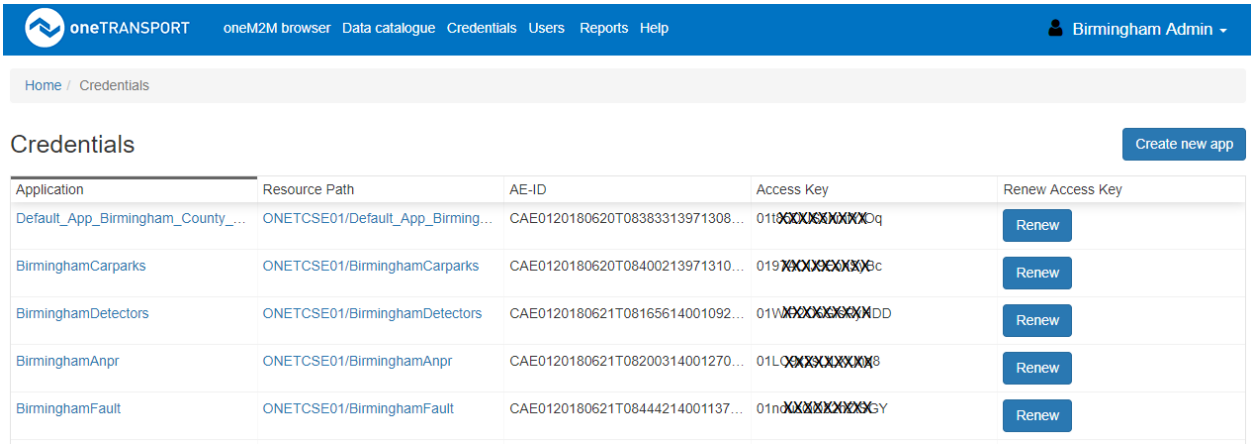
To utilize any of the oneTRANSPORT API's, you will need to authenticate each call to the API with an **AE-ID** and an **Access Key**. Both of these values are created when an Application is added to the portal. An Application within oneTRANSPORT is both the Application that publishes data, and the Application that consumes data. In this instance we refer to Application as that which is consuming shared data of another Application.

Authentication is set within headers sent to the oneTRANSPORT API:

```
X-M2M-Origin: Your-applications-AE-ID
```

```
Authorization: Bearer Your-applications-access-key
```

These credentials can be found by browsing to the **Credentials** page in oneTRANSPORT:



Creating a Container

A Container is an entity that is associated to an object within your application. You are able to create Containers within Containers for the purpose of grouping your data. For example, a number of Traffic Lights may be grouped per road, and then roads grouped per area. This would allow data consumers to discover your data entities.

Containers are created by posting a web service call to the oneTRANSPORT App. The URL for the post is defined within the Getting Started section on the portal and will be like:

```
https://onem2m.onetransport.io/ONETCSE01/your-application
```

Please see the Addressing Resources section later in this document for more information.

Request Headers

Please note that Request Headers are case sensitive, and oneM2M headers **must be sent in upper case**.

`Content-Type: application/vnd.onem2m-res+json; ty=3`

Use the MIME type of the request body, and set the ty parameter to be 3 to indicate that the body represents a Container

`X-M2M-RI: your-request-identifier`

A request identifier. This identifier can be used by your application to match responses with requests.

`X-M2M-Origin: Your-applications-AE-ID`

Your App's AE-ID

`Authorization: Bearer Your-applications-access-key`

Your App's access token - it must match the AE-ID

Request body

The request body should contain a JSON or XML representation of the Container resource.

For a create request it must contain the following attributes:

`rn` (resourceName)

A name for the Container.

`acpi` (accessControlPolicyIds)

A list of Access Control Privileges which will be applied to this Container. The list should contain the ACP that has been defined for your Application, which is available by browsing to the Application in the resource tree browser. Using this ACP will allow you to publish your Container on the Developer Portal

You will find an example of the JSON within the Getting Started section, and you will need to ensure that you use the correct AE-ID and ACPI for your App. An example of the JSON is below:

```
{
    "m2m:cnt": {
        "rn": "SIRI",
        "acpi": ["your-applications-acp"],
        "lbl": "Chordant City Bus SIRI Feeds"
    }
}
```

Web service response

If the web service call has been successful, you shall receive a HTTP response code 201 Created. The response shall contain the following oneM2M headers:

`X-M2M-RI: your-request-identifier`

This is the same as the request header.

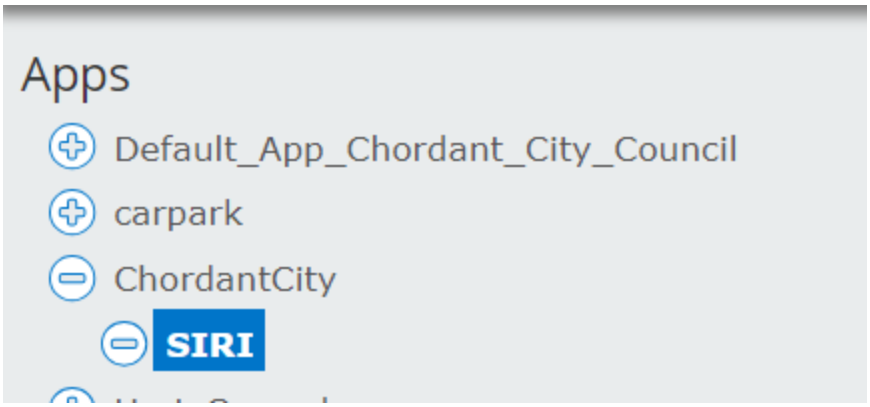
`X-M2M-RSC: 2001`

This is the response status code returned by oneM2M. Further information on this can be found later in this document.

The response from the web service shall contain output from oneM2M such as below. The content contains a name value pairs using oneM2M nomenclature. An explanation of the nomenclature is found later in this document.

```
{
  "m2m:cnt": {
    "cbs": 0,
    "cni": 0,
    "ct": "20170406T114103",
    "et": "99991231T235959",
    "lt": "20170406T114103",
    "mbs": 60000000,
    "mia": 1600,
    "mni": 10000,
    "pi": "C-Y249RW1wdHlBcHAsb3U9cm9vdA",
    "ri": "cnt20170406T11410313999393354933760",
    "st": 0,
    "ty": 3
  }
}
```

Once you have received a successful response from the web service, the Container shall be visible in the browser tree:



Sending Data to a Container

Within oneTRANSPORT, data is organized into Containers. A Container can, for example, be a specific sensor, or a specific location such as a car park. Containers are defined by the publisher of the data, therefore you need to be able to discover and understand the data that you wish to consume. Data transactions are stored within oneTRANSPORT as Content Instances. A single Content Instance contains a single piece of data sent by the publishing application.

If you are a publisher of data, then you are able to create Content Instances to store your data within oneTRANSPORT. Content Instances are created by posting a web service call to the oneTRANSPORT App. The URL for the post is defined within the Getting Started section, and will be similar to:

```
https://onem2m.onetransport.io/ONETCSE01/ChordantCity/SIRI/Green/Stops/SciencePark
```

Please see the Addressing Resources section later in this document for more information.

Request Headers

Please note that Request Headers are case sensitive, and oneM2M headers must be sent in upper case.

`Content-Type: application/vnd.onem2m-res+json; ty=4`

Use the MIME type of the request body, and set the `ty` parameter to be `4` to indicate that the body represents a Content Instance

`X-M2M-RI: your-request-identifier`

A request identifier. This identifier can be used by your application to match responses with requests. This identifier

`X-M2M-Origin: Your-applications-AE-ID`

Your App's AE-ID

`Authorization: Bearer Your-applications-access-key`

Your App's access token - it must match the AE-ID

Request body

The request body should contain a JSON or XML representation of the Content Instance resource.

For a create request it must contain the following attributes:

`cnf` (contentType)

A string which consists of the content type (e.g. `text/plain` or `application/json`) followed by an encoding type.

Encoding type can be either:

- `0` - Plain - no transfer encoding is applied
- `1` - Base64 encoding is applied on string data
- `2` - Base64 encoding is applied on binary data

`con` (content)

The content itself

An example of the JSON is below. This example shows JSON being set within the Content Instance:

```
{
   "m2m:cin": {
      "cnf": "application/json:0",
      "con": {
         "data": "{'bus 1': {'id': 'bus001', 'timestamp': [ '20170406T114103'], 'state': [ 'on time']}}"
```

```
        }
    }
}
```

The following example shows some text being set as the data for the Content Instance:

```
{
   "m2m:cin": {
      "cnf": "text/plain:0",
      "con": "Bus is on time"
   }
}
```

Web service response

If the web service call has been successful, you shall receive a HTTP response code 201 Created. The response shall contain the following oneM2M headers:

`X-M2M-RI: your-request-identifier`

This is the same as the request header.

`X-M2M-RSC: 2001`

This is the response status code returned by oneM2M. Further information on this can be found later in this document.

Once you have received a successful response from the web service, the Content Instance shall be visible in the Container as the Latest Content Instance.

Retrieving Data from a Container

Within oneTRANSPORT, data is organized into Containers. A Container can, for example, be a specific sensor, or a specific location such as a bus stop. Containers are defined by the publisher of data and therefore you need to be able to discover and understand the data that you wish to consume. Data transactions are stored within oneTRANSPORT as Content Instances. A single Content Instance contains a single piece of data sent by the publishing application.

We read Content Instances from oneTRANSPORT, by sending a web service call asking for the latest Content Instance. To do this you will need to make a `GET` request to the URL of the Application and its Container, for example:

```
https://onem2m.onetransport.io/ONETCSE01/ChordantCity/SIRI/Green/Stops/SciencePark/la
```

You will notice the end of this request contains a virtual resource named **la** which denotes the **la**test Content Instance. You are also able to return the **ol**dest Content Instance by specifying **ol**.

Request headers

```
X-M2M-RI: your-request-identifier

X-M2M-Origin: Your-applications-AE-ID

Authorization: Bearer Your-applications-access-key

Accept: application/json
```

If the Accept header is set as above, then the response shall be in JSON format. If the Accept header is excluded, then the standard response shall be in XML format.

HTTP Response

The request will return a response indicating the status of the request. Successful requests will contain a `X-M2M-RSC: 2000` header. The body of the response shall contain the oneM2M output. The "con" attribute will contain the data for this Content Instance.

```
HTTP/1.1 200 Content
Server: nginx/1.9.10
Date: Fri, 1 January 2018 10:11:35 GMT
Content-Type: application/vnd.onem2m-res+json
Content-Length: 235
Connection: keep-alive
X-M2M-RI: your-request-identifier
X-M2M-RSC: 2000
{
```

```
    "m2m:cin": {
        "cnf": "text/plain:0",
        "con": "Bus is on time",
        "cs": 14,
        "ct": "20190425T153330",
        "et": "99991130T235959",
        "lt": "20190425T153330",
        "pi": "cnt20190425T151512139794793355008 2187_cse01",
        "ri": "cin20190425T153330140020857939712 36_cse01",
        "rn": "cin20190425T153330140020857939712 35_cse01",
        "st": 2,
        "ty": 4
    }
}
```


Subscribing to a Container

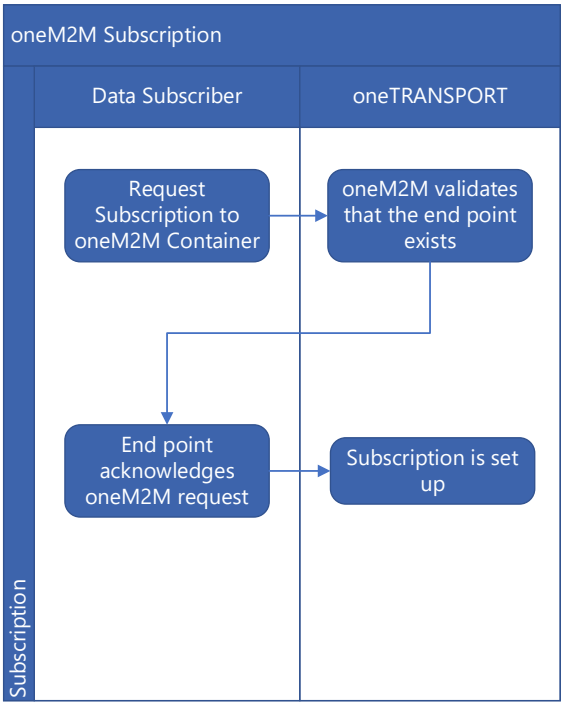
To retrieve Content Instances from a Container, you can query the Container (this is covered later in this document). You are also able to Subscribe to a Container. When you Subscribe to a Container you provide oneTRANSPORT with the end point that you would like to use to receive notifications that new data has been received by the Container. You will then be forwarded the data.

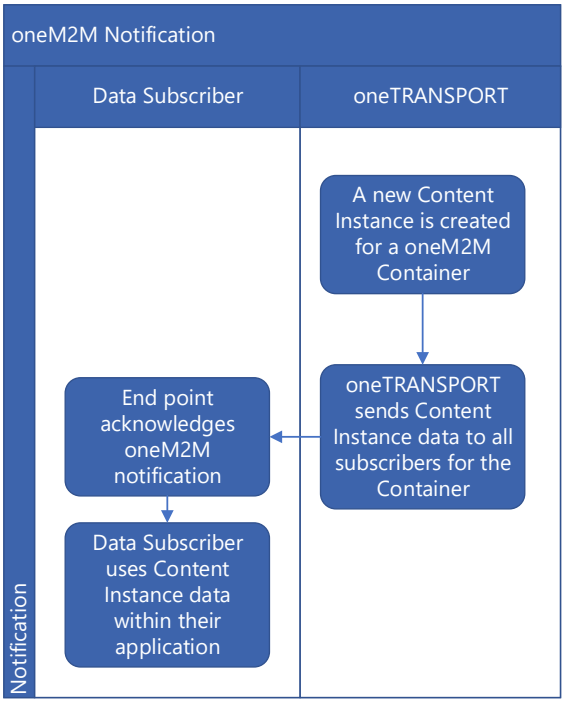
To receive a notification when new content is added you will need to create a Subscription resource in the Container where the new content is added. Your application will need to expose a HTTP endpoint capable of validating a subscription and receiving notifications.

The HTTP endpoint should handle three types of incoming request:

- When a oneM2M subscription resource is created, the oneM2M server will validate the HTTP endpoint. It does this by sending a subscription verification request to the notification endpoint.
- When new content is added to the target data Container the HTTP endpoint will need to acknowledge the new content

- Finally, when the subscription is deleted the HTTP endpoint will be notified, and it should acknowledge that the subscription is no longer available. The oneM2M server will no longer send any further notifications





The following rules must be strictly followed to ensure that your subscription request is accepted:

- The oneM2M headers must be returned by your web service is upper case.
- The header `X-M2M-RI` must be sent as a response header, and the contents of the header must be exactly the same as the `X-M2M-RI` request header.
- Your end point must return the value 2001 in the response header `X-M2M-RSC` to indicate to oneTRANSPORT that the request for the subscription has been accepted.

Create the subscription

As your end point shall be used for both the initial subscription request and for notifications of new Content Instance, you must implement a check in your endpoint for the contents of the HTTP request body. The request body will contain the `vrq` attribute upon the initial subscription.

To make the subscription request, you must make a POST request to the URL of the Container. The url will be shown within the portal and will be similar to the following:

```
https://onem2m.onetransport.io/ONETCSE01/ChordantCity/SIRI/Green/Stops/SciencePark
```

Request Headers

Please note that Request Headers are case sensitive, and oneM2M headers must be sent in upper case.

`Content-Type: application/vnd.onem2m-res+json; ty=23`

Use the MIME type of the request body, and set the `ty` parameter to be `23` to indicate that the body represents a subscription request

`X-M2M-RI: your-request-identifier`

A request identifier. This identifier can be used by your application to match responses with requests.

`X-M2M-Origin: Your-applications-AE-ID`

Your App's AE-ID

`Authorization: Bearer Your-applications-access-key`

Your App's access token - it must match the AE-ID

Request Body

The request body should contain a JSON or XML representation of the Content Instance resource.

For a create request it must contain the following attributes:

`enc` (eventNotificationCriteria)

`net` (notificationEventType)

The conditions which need to be met for your HTTP endpoint to be notified. For notification of new content then `3` should be used, as new Content Instances are being created as children of a Container.

The other options are:

- `1` - Update of resource

`2` - Delete of resource

`3` - Create of direct child resource

`4` - Delete of direct child resource

`rn` (resourceName)

A name for the subscription.

`acpi` (accessControlPolicyIds)

A list of Access Control Privileges which will be applied to the subscription. The list should contain the ACP that has been defined for your Application, which is available by browsing to the Application in the resource tree browser. Using this ACP will allow you to identify your Apps subscriptions.

`nu` (notificationURI)

A URI to your HTTP endpoint. It must include the scheme (HTTP / HTTPS) and the port number.

You will find an example of the JSON within the Getting Started section, and you will need to ensure that you use the correct AE-ID and ACPI for your App. An example of the JSON is below:

```
{
    "m2m:sub": {
        "rn": "ChordantCity-SIRI-Green-Stops-SciencePark-Sub",
        "enc": {
            "net": ["3"]
        },
        "acpi": ["your-applications-acp"],
        "nu": ["http://my.hostname:8080/my-endpoint"]
    }
}
```

Web service response

If the web service call has been successful, you shall receive a HTTP response code 201 Created. The response shall contain the following oneM2M headers:

`X-M2M-RI: your-request-identifier`

This is the same as the request header.

`X-M2M-RSC: 2001`

This is the response status code returned by oneM2M. Further information on response codes can be found later in this document.

Receiving Notifications from a Container

Once a subscription to a Container is requested, the End Point that was used to validate the subscription is also used to receive notifications of new Content Instances for the Container.

The below pseudo code should be followed within your End Point logic:

```
IF response body contains vrq THEN
        // Subscription Validation.
        X-M2M-RI: [Request Header X-M2M-RI]
        X-M2M-RSC: 2001
        HTTP Status Code: 201
ELSE
        IF request verb is DELETE THEN
                // Remove subscription.
                X-M2M-RI: [Request Header X-M2M-RI]
                X-M2M-RSC: 2000
                HTTP Status Code: 200
        ELSE
                // New Content Notification.
                X-M2M-RI: [Request Header X-M2M-RI]
                X-M2M-RSC: 2000
                HTTP Status Code: 200
        END IF
```

```
END IF
```

For new content and delete notifications the HTTP endpoint should return HTTP status code `200 OK`, and the oneM2M response code of `2000` in the `X-M2M-RSC` response header.

The HTTP request body of the notification will contain a representation of the resource encoded as a string. The request body will either be in JSON or XML, depending on what was used by the AE that created the initial Content Instance. It will also contain the resource ID of the subscription that resulted in the notification.

Deleting a Container

Containers are deleted by using the `DELETE` verb via web service call to the oneTRANSPORT App. The URL for the delete will be like:

```
https://onem2m.onetransport.io/ONETCSE01/your-application
```

Please see the Addressing Resources section later in this document for more information.

Request Headers

Please note that Request Headers are case sensitive, and oneM2M headers **must be sent in upper case**.

`X-M2M-Origin: Your-applications-AE-ID`

Your App's AE-ID

`Authorization: Bearer Your-applications-access-key`

Your App's access token - it must match the AE-ID

Request body

No request body is required to be sent to for a deletion.

Web service response

If the web service call has been successful, you shall receive a HTTP response code 200 OK. The response shall contain the following oneM2M headers:

`X-M2M-RI: your-request-identifier`

This is the same as the request header.

`X-M2M-RSC: 2002`

This is the response status code returned by oneM2M. Further information on this can be found later in this document.

The response from the web service shall contain output from oneM2M such as below. The content contains a name value pairs using oneM2M nomenclature. An explanation of the nomenclature is found later in this document.

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="StationWay">
    <ty>3</ty>
    <ri>cnt20190425T151521139794549966592226_cse01</ri>
    <pi>cnt20190425T151452139794482824960215 4_cse01</pi>
    <ct>20190425T151521</ct>
    <lt>20190425T151521</lt>
    <lbl>label0 label1</lbl>
    <acpi>acp20190425T143855139794398897920207 1_cse01</acpi>
    <et>99991130T235959</et>
    <st>0</st>
    <mni>10000</mni>
    <mbs>60000000</mbs>
    <mia>1600</mia>
    <cni>0</cni>
    <cbs>0</cbs>
    <disr>false</disr>
</m2m:cnt>
```

Once you have received a successful response from the web service, the Container shall be deleted.

Appendix

Authorization

Requests to the oneTRANSPORT use HTTP token-based authorization. After creating an application on the Developer Portal you will be provided with two peices of information:

The AE-ID for the Application you have created and,

The token that is associated with the Application

The AE-ID should be set in the `X-M2M-Origin` header. The `Authorization` header should be set to `Bearer <Token>`, where `<token>` is the value obtained from the Developer Portal.

These two headers are used in every request to oneTRANSPORT.

Common Resource Attributes

`rn` - Resource Name

`ri` - Resource Identifier

`lbl` - An array of labels that can be used to identify the data

Common Request Headers

`Accept`

The `Accept` header should be set to either:

- `application/json` or `application/vnd.onem2m-res+json` to receive JSON
- `application/vnd.onem2m-res+xml` to receive XML

`X-M2M-Origin`

The `X-M2M-Origin` header contains the AE-ID

`X-M2M-RI`

The `X-M2M-RI` header contains a request identifier. The request identifier can be used by the Application Entity to match responses to corresponding requests.

Common Response Headers

`Content-Type`

The `Content-Type` response header should match the `Accept` request header. It indicates what type of content is within the response body.

`X-M2M-RI`

The `X-M2M-RI` response header should match the corresponding request header.

`X-M2M-RSC` - oneM2M Response Status Code

`Content-Location`

On a create request the `Content-Location` header contains a URL to the Resource ID.

Addressing Resources

Resources can be addressed by using either the Resource ID or the Resource name. The Resource ID is an identity that is generated by the server at the time of creation. The Resource Name is created by the application in the initial request. When addressing a Resource using the name, each of names of the parent Resources need to be included in the path. Alternatively, when addressing by Resource ID, only the CSE ID and the Resource ID are required.

Example of Resource ID address:

```
~/ONET-CSE-01/CAE0120180613T083238139764871030528l3087_cse01
```

Example of Resource Name address:

```
/ONETCSE01/ChordantCity/SIRI/Green/Stops/SciencePark
```

Response Status Codes

Code	Description
1000	ACCEPTED
2000	OK
2001	CREATED
2002	DELETED
2004	CHANGED

Code	Description
4000	BAD REQUEST
4004	NOT FOUND
4005	OPERATION NOT ALLOWED
4008	REQUEST TIMEOUT
4101	SUBSCRIPTION CREATOR HAS NO PRIVILEGE
4102	CONTENTS UNACCEPTABLE
4103	ACCESS DENIED
4104	GROUP REQUEST IDENTIFIER EXISTS
4105	CONFLICT
5000	INTERNAL SERVER ERROR
5001	NOT IMPLEMENTED
5103	TARGET NOT REACHABLE
5105	NO PRIVILEGE
5106	ALREADY EXISTS
5203	TARGET NOT SUBSCRIBABLE
5204	SUBSCRIPTION VERIFICATION INITIATION FAILED
5205	SUBSCRIPTION HAS NO PRIVILEGE
5206	NON BLOCKING REQUEST NOT SUPPORTED
5207	NOT ACCEPTABLE
6003	EXTERNAL OBJECT NOT REACHABLE

Application Entity

Application Entity Attributes

`api` - Application Identifier

`rr` - Request Reachability

If set to `true` then the CSE will attempt to make a request to the `poa` (point of access) URI when the AE is first created.

`acpi` - Access Control Policies

A list of access control policies that define the rights and privileges that other AEs have over the content. A default ACP is provided, `ONETCSE01Acp`, that provides read-write permission to the originating AE, and read-only to all other AEs.

`poa` - Point of access

A URL that will be used by the CSE to verify and send subscription notifications to.

Create

oneM2M supports the creation of Application Entities with HTTP POST requests, but this functionality has been disabled for the oneTRANSPORT platform. Please create Applications using the Developer Portal.

Retrieve

Example request to retrieve an Application Entity:

```
$ curl -v -H"Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: ae_retrieve" \
     -H "X-M2M-Origin: MyAEID " \
     "https://onem2m.onetransport.io/ONETCSE01/ChordantCity"
```

Response when retrieving an Application Entity:

```
> GET /ONETCSE01/MyApp HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> X-M2M-Origin: MyAEID
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: ae_retrieve
>
< HTTP/1.1 200 Content
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 08:39:50 GMT
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 282
< Connection: keep-alive
< X-M2M-RI: ae_retrieve
< X-M2M-RSC: 2000
<
{
```

```
  "m2m:ae": {
    "acpi": [
      "your-applications-acp"
    ],
    "aei": "MyToken ",
    "api": "MyToken ",
    "apn": "ChordantCity",
    "ct": "20161013T084121",
    "et": "20200101T000000",
    "lt": "20161013T084126",
    "pi": "ONET-CSE-01",
    "ri": "C-Y249TXlBcHAsb3U9cm9vdA",
    "rn": "ChordantCity",
    "rr": true,
    "ty": 2
  }
}
```

Update

Example HTTP request body when updating an Application Entity:

```
{
    "ae": {
        "apn": "MyAppName"
    }
}
```

Example HTTP request to update an Application Entity:

```
$ curl -v -X PUT \
     -H "Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: ae_update" \
```

```
     -H "X-M2M-Origin: MyAEID " \
     -H "Content-Type: application/json" \
     -d @ae_update.json "https://onem2m.onetransport.io/ONETCSE01/MyAEName
"
```

Example HTTP request and response:

```
> PUT /ONETCSE01/MyAEName HTTP/1.1
> Host: onem2m.onetransport.io
> Authorization: Basic b2dyaWZmaW46dGVzdA==
> User-Agent: curl/7.47.0
> Accept: application/json
> X-M2M-RI: ae_update
> X-M2M-Origin: MyAEID
> Content-Type: application/json
> Content-Length: 44
>
{
  "ae": {
    "apn": "MyAppName"
  }
}
< HTTP/1.1 200 Changed
< Content-Type: application/vnd.onem2m-res+json
< X-M2M-RI: ae_update
< X-M2M-RSC: 2004
< Content-Length: 54
<
{
  "m2m:ae": {
    "apn": "MyAppName",
    "lt": "20160922T215805"
  }
```

```
}
```

Discovery

```
$ curl -v -H"Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: ae-discovery" \
     -H "X-M2M-Origin: MyAEID " \
     "https://onem2m.onetransport.io/ONETCSE01?fu=1&ty=2"
```

Transcript of HTTP request and response to discover Application Entities:

```
> GET /ONETCSE01?fu=1&ty=2 HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> Accept: application/json
> X-M2M-RI: ae_discovery
> X-M2M-Origin: MyAEID
>
< HTTP/1.1 200 Content
< Content-Type: application/vnd.onem2m-res+json
< X-M2M-RI: ae_discovery
< X-M2M-RSC: 2000
< Content-Length: 49
<
{
  "m2m:uril": [
    "/ONET-CSE-01/ONETCSE01/MyAEName"
  ]
}
```

Delete

oneM2M supports the deletion of Application Entities, but this functionality has been disabled for the oneTRANSPORT platform.

Containers

Attributes

All of the Common Resource Attributes

- `pi` - Parent Resource Identifier

The Resource Identifier of the parent Resource in the hierarchy. This can either be an Application Entity or another Container.

- `cbs` - Current Byte Size
- `cni` - Current Number of Content Instances

A count for the number of Content Instances currently within the Container

- `mni` - Maximum Number of Content Instances

When the number of Content Instances in the Container reach this value, then the CSE will expire them.

Create

Request headers

```
Content-Type: application/vnd.onem2m-res+json;ty=3
```

The `Content-Type` will need to include the MIME type of the request body content and the enumeration type of oneM2M resource created as the `ty` parameter. Containers have the enumeration type value of `3`

`X-M2M-RI` and `X-M2M-Origin` - see Common Request Headers

Request body

The request body should contain a JSON or XML representation of the Container resource.

It *must* contain the following attributes:

`rn` - Resource Name

Request path

A Container can be placed within the CSE structure in an Application Entity or existing Container. It is not possible to create a Container under the CSE base.

Example:

The file `cnt_create.json` contains a JSON representation of a Container:

```
{
    "cnt": {
        "rn": "<<cnt-name>>",
        "lbl": ["label0", "label1", "label3"]
    }
}
```

Example request to create a Container:

```
curl -v  -X POST -H"Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: cnt-create" \
     -H "X-M2M-Origin: MyAEID " \
     -H "Content-Type: application/vnd.onem2m-res+json; ty=3" \
     -d @cnt_create.json "https://onem2m.onetransport.io/ONETCSE01/MyAENam
e"
```

Transcript of HTTP request and response to create a Container:

```
> POST /ONETCSE01/MyApp4 HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> X-M2M-Origin: MyAEID
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cnt-create
> Content-Type: application/vnd.onem2m-res+json; ty=3
> Content-Length: 92
>
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 10:10:15 GMT
< Content-Type: application/vnd.onem2m-res+json
```

```
< Content-Length: 200
< Connection: keep-alive
< X-M2M-RI: cnt_create
< X-M2M-RSC: 2001
< Content-Location: /ONET-CSE-01/cnt_20161014T101014_6512689
<
{
  "m2m:cnt": {
    "cbs": 0,
    "cni": 0,
    "ct": "20161014T101014",
    "et": "99991231T235959",
    "lt": "20161014T101014",
    "pi": "C-Y249TXlBcHA0LG91PXJvb3Q",
    "ri": "cnt_20161014T101014_6512689",
    "rn": "MyContainer",
    "st": 0,
    "ty": 3
  }
}
```

Retrieve

Example request to retrieve a Container:

```
curl -v -H"Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: cnt-retrieve" \
     -H "X-M2M-Origin: MyAEID " \
     "https://onem2m.onetransport.io/ONETCSE01/MyAEName/MyContainer"
```

Transcript of HTTP request and response to retrieve a Container:

```
> GET /ONETCSE01/MyApp4/MyContainer HTTP/1.1
```

```
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> X-M2M-Origin: MyAEID
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cnt_retrieve
>
< HTTP/1.1 200 Content
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 10:11:35 GMT
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 235
< Connection: keep-alive
< X-M2M-RI: cnt_retrieve
< X-M2M-RSC: 2000
<
{
  "m2m:cnt": {
    "cbs": 0,
    "cni": 0,
    "ct": "20161014T101014",
    "et": "99991231T235959",
    "lbl": [
      "label0",
      "label1",
      "label3"
    ],
    "lt": "20161014T101014",
    "pi": "C-Y249TXlBcHA0LG91PXJvb3Q",
    "ri": "cnt_20161014T101014_6512689",
    "rn": "MyContainer",
    "st": 0,
```

```
    "ty": 3
  }
}
```

Update

To modify the attributes of a Container a `PUT` request can be made to the Container path.

The file `cnt_update.json` contains a JSON representation of a Container:

```
{
    "cnt": {
        "lbl": ["label0", "label1"]
    }
}
```

Example request to update an existing Container:

```
curl -v -X PUT \
     -H "Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: cnt-update" \
     -H "X-M2M-Origin: MyAEID " \
     -H "Content-Type: application/vnd.onem2m-res+json" \
     -d @cnt_update.json "https:// onem2m.onetransport.io/ONETCSE01/MyApp4
/MyContainer"
```

Transcript of HTTP request and response to update a Container:

```
> PUT /ONETCSE01/MyApp4/MyContainer HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cnt-update
> X-M2M-Origin: MyAEID
```

```
> Content-Type: application/vnd.onem2m-res+json
> Content-Length: 74
>
* upload completely sent off: 74 out of 74 bytes
< HTTP/1.1 200 Changed
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 11:04:44 GMT
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 70
< Connection: keep-alive
< X-M2M-RI: cnt-update
< X-M2M-RSC: 2004
<
{
  "m2m:cnt": {
    "lbl": [
      "label0",
      "label1"
    ],
    "lt": "20161014T110444",
    "st": 3
  }
}
```

Discovery

Discover data Containers created between 10:25 and 10:30 on the 14th October 2016:

```
curl -v -H"Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: cnt-discovery" \
     -H "X-M2M-Origin: MyAEID " \
     "https:// onem2m.onetransport.io/ONETCSE01?fu=1&ty=3&cra=20161014T102500&crb=20161014T103000"
```

Transcript of HTTP request and response to discover Containers:

```
> GET /ONETCSE01?fu=1&ty=3&cra=20161014T102500&crb=20161014T103000 HTTP/1.
1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> X-M2M-Origin: MyAEID
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cnt-discovery
> X-M2M-Origin: MyAEID
>
< HTTP/1.1 200 Content
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 10:33:15 GMT
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 168
< Connection: keep-alive
< X-M2M-RI: cnt_discovery
< X-M2M-RSC: 2000
<
{
  "m2m:uril": [
    "/ONET-CSE-01/ONETCSE01/Hertfordshire/v2.0/RoadWorks/GUID4061122",
    "/ONET-CSE-01/ONETCSE01/MyApp4/MyContainer/cnt19850823T094554493638354
139773712779008"
  ]
}
```

It is also possible to discover Containers that belong to any AE by including the AE name in the path:

```
curl -v -H"Authorization: Bearer 01ZU2D2nCo7HDlrv" \
     -H "Accept: application/json" \
```

```
     -H "X-M2M-RI: cnt-discovery-ae" \
     -H "X-M2M-Origin: MyAEID " \
     "https://onem2m.onetransport.io/ONETCSE01/MyApp4?fu=1&ty=3&cra=201610
14T102500&crb=20161014T103000"
```

Transcript of HTTP request and response to discover Containers for `MyApp4`:

```
> GET /ONETCSE01/MyApp4?fu=1&ty=3&cra=20161014T102500&crb=20161014T103000
HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cnt-discovery-ae
> X-M2M-Origin: MyAEID
>
< HTTP/1.1 200 Content
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 10:45:28 GMT
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 102
< Connection: keep-alive
< X-M2M-RI: cnt-discovery-ae
< X-M2M-RSC: 2000
<
{
  "m2m:uril": [
    "/ONET-CSE-01/ONETCSE01/MyApp4/MyContainer/cnt19850823T094554493638354
139773712779008"
  ]
}
```

Delete

Example request to delete a Container:

```
curl -v -X DELETE \
     -H "Authorization: Bearer MyToken" \
     -H "X-M2M-RI: cnt-delete" \
     -H "X-M2M-Origin: MyAEID" \
     -H "Content-Type: application/vnd.onem2m-res+json" \
     -d " https://onem2m.onetransport.io/ONETCSE01/ChordantCity/SIRI/Green
/Stops/StationWay”
```

Transcript of HTTP request and response to retrieve a Container:

```
> GET /ONETCSE01/MyApp4/MyContainer HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> X-M2M-Origin: MyAEID
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cnt_delete
>
< HTTP/1.1 200 Content
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 10:11:35 GMT
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 235
< Connection: keep-alive
< X-M2M-RI: cnt_delete
< X-M2M-RSC: 2002
<
<?xml version="1.0" encoding="UTF-8"?>
<m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="StationWay">
    <ty>3</ty>
    <ri>cnt20190425T151521139794549966592226_cse01</ri>
```

```
    <pi>cnt20190425T151452139794482824960215 4_cse01</pi>
    <ct>20190425T151521</ct>
    <lt>20190425T151521</lt>
    <lbl>label0 label1</lbl>
    <acpi>acp20190425T143855139794398897920207 1_cse01</acpi>
    <et>99991130T235959</et>
    <st>0</st>
    <mni>10000</mni>
    <mbs>60000000</mbs>
    <mia>1600</mia>
    <cni>0</cni>
    <cbs>0</cbs>
    <disr>false</disr>
</m2m:cnt>
```

Content Instance

It is not possible to **UPDATE** an existing Content Instance.

Create

The file `cin_create.json` contains a JSON representation of a Content Instance:

```
{
  "cin": {
    "rn": "<<cin-name>>",
    "cnf": "text/plain:0",
    "con": "Hello World"
  }
}
```

Example request to create a Content Instance:

```
curl -v -X POST \
```

```
     -H "Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: cin-create" \
     -H "X-M2M-Origin: MyAEID " \
     -H "Content-Type: application/vnd.onem2m-res+json; ty=4" \
     -d @cin_create.json "https://onem2m.onetransport.io/ONETCSE01/MyApp4/
MyContainer"
```

Transcript of HTTP request and response to create a Content Instance:

```
> POST /ONETCSE01/MyApp4/MyContainer HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cin-create
> X-M2M-Origin: MyAEID
> Content-Type: application/vnd.onem2m-res+json; ty=4
> Content-Length: 123
>
< HTTP/1.1 201 Created
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 10:59:20 GMT
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 200
< Connection: keep-alive
< X-M2M-RI: cin-create
< X-M2M-RSC: 2001
<
{
  "m2m:cin": {
    "cs": 11,
    "ct": "20161014T105920",
```

```
    "et": "99991231T235959",
    "lt": "20161014T105920",
    "pi": "cnt_20161014T101014_6512689",
    "ri": "cin_20161014T105920_6592167",
    "rn": "MyContentInstance",
    "st": 2,
    "ty": 4
  }
}
```

Latest Content Instance

To read the last Content Instance added to a Container retrieve the `la` virtual resource:

```
curl -v -H "Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: cin-latest" \
     -H "X-M2M-Origin: MyAEID " \
     "https://onem2m.onetransport.io/ONETCSE01/MyApp4/MyContainer/la"
```

Transcript of HTTP request and response to retrieve the latest Content Instance for a Container:

```
> GET /ONETCSE01/MyApp4/MyContainer/la HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cin-latest
> X-M2M-Origin: MyAEID
>
< HTTP/1.1 200 Content
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 11:08:32 GMT
```

```
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 241
< Connection: keep-alive
< X-M2M-RI: cin-latest
< X-M2M-RSC: 2000
<
{
  "m2m:cin": {
    "cnf": "text/plain:0",
    "con": "Hello World",
    "cs": 11,
    "ct": "20161014T105920",
    "et": "99991231T235959",
    "lt": "20161014T105920",
    "pi": "cnt_20161014T101014_6512689",
    "ri": "cin_20161014T105920_6592167",
    "rn": "MyContentInstance",
    "st": 2,
    "ty": 4
  }
}
```

Oldest Content Instance

To read the oldest Content Instance for a Container retrieve the `ol` virtual resource:

```
curl -v -H "Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: cin-latest" \
     -H "X-M2M-Origin: MyAEID " \
     "https://onem2m.onetransport.io/ONETCSE01/MyApp4/MyContainer/ol"
```

Transcript of HTTP request and response to retrieve the `ol` virtual resource:

```
> GET /ONETCSE01/MyApp4/MyContainer/ol HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> Authorization: Bearer MyToken
> Accept: application/json
> X-M2M-RI: cin-latest
> X-M2M-Origin: MyAEID
>
< HTTP/1.1 200 Content
< Server: nginx/1.9.10
< Date: Fri, 14 Oct 2016 11:11:23 GMT
< Content-Type: application/vnd.onem2m-res+json
< Content-Length: 241
< Connection: keep-alive
< X-M2M-RI: cin-latest
< X-M2M-RSC: 2000
<
{
  "m2m:cin": {
    "cnf": "text/plain:0",
    "con": "Hello World",
    "cs": 11,
    "ct": "20161014T105920",
    "et": "99991231T235959",
    "lt": "20161014T105920",
    "pi": "cnt_20161014T101014_6512689",
    "ri": "cin_20161014T105920_6592167",
    "rn": "MyContentInstance",
    "st": 2,
    "ty": 4
  }
```

```
}
```

Subscription

Create

The file `sub_create.json` contains a JSON representation of a Subscription:

```
{
  "m2m:sub": {
    "enc": {
      "net": ["3"]
    },
    "nu": ["<<ae-id>>"]
  }
}
```

Request to create a subscription:

```
curl -v -X POST \
     -H "Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: sub_create" \
     -H "X-M2M-Origin: MyAEID " \
     -H "Content-Type: application/vnd.onem2m-res+json; ty=23" \
     -d @sub_create.json "https://onem2m.onetransport.io/ONETCSE01/MyAENam
e/MyContainer"
```

Response from Subscription **CREATE**:

```
HTTP/1.1 201 Created
Content-Type: application/vnd.onem2m-res+json
X-M2M-RI: sub_create
X-M2M-RSC: 2001
Content-Location: /ONET-CSE-01/sub_20160919T080026_1
Content-Length: 199
```

```
{
  "m2m:sub": {
    "ct": "20160919T080026",
    "et": "99991231T235959",
    "lt": "20160919T080026",
    "pi": "cnt_20160919T080017_0",
    "ri": "sub_20160919T080026_1",
    "rn": "sub19850418T1052284826695481406843864942O8",
    "ty": 23
  }
}
```


Notifications

When content is added to the subscribed Containers the AE receives notifications from the CSE to its defined point-of-access.

The URL path of the point-of-access will need to correspond to the AE ID. For example, if the AE has the ID “MyAE” then the path will be “/CMyAE”.

The HTTP server will need to respond to the notification with a valid oneM2M response, including the `X-M2M-RI` and `X-M2M-RSC` response headers. The `X-M2M-RI` header should value should duplicate the value provided by the request header of the same name. The `X-M2M-RSC` header describes the status of the operation. In this case the value `2001` should be returned to indicate that content has been created.

The HTTP request body received by the point-of-access will contain a HTTP representation of the oneM2M notification resource. Whether or not the HTTP request is XML or JSON is determined by the originator of the content being created. The majority of oneTRANSPORT AEs use JSON - so the HTTP request body will be in JSON format.

The notification resource contains a `rep` attribute which contains a representation of the oneM2M resource which has been modified. This will be a Content Instance, and the JSON will be URI encoded within a string. The Content Instance itself has a `con` attribute which contains the actual content required by the journey time engine.

If multiple subscriptions are created, then the point-of-access will be receiving notifications for multiple Containers. The AE

will need to use the Resource ID of the Subscription to correlate which Container the notification is related to.

```
POST /C-MyAE HTTP/1.1
Content-Type: application/vnd.onem2m-ntfy+json
Content-Length: 373
Accept: application/vnd.onem2m-res+json
X-M2M-RI: my_value_must_match
X-M2M-Origin: https://onem2m.onetransport.io/ONET-CSE-01


{
  "m2m:sgn": {
    "nev": {
      "net": ["3"],
      "rep": "{\"m2m:cin\":{\"cbs\":0,\"cnf\":\"text/plain\",\"cni\":0,\"con\":\"JSON HERE\",\"cs\":5,\"ct\":\"20160620T220616\",\"et\":\"20160624T092616\",\"lt\":\"20160620T220616\",\"pi\":\"cnt_20160620T220615_97\",\"ri\":\"cin_20160620T220616_99\",\"rn\":\"cin20160620T220616140276565894912\",\"st\":2,\"ty\":4}}\n"
    },
    "sur": "SUBSCRIPTION_ID"
  }
}
```

The attribute `rep` is a representation of the resource subscribed to. In this case it is a Content Instance. The value of `JSON HERE` will be an escaped-JSON representation of the content.

The `/CMyAE` endpoint should return:

```
HTTP/1.1 201 Created
X-M2M-RI: my_value_must_match
X-M2M-RSC: 2001
Content-Length: 0
```

Browse Resource Tree

Request to list the child nodes of the CSE base:

```
curl -v -X GET \
     -H "Authorization: Bearer MyToken" \
     -H "Accept: application/json" \
     -H "X-M2M-RI: sub_create" \
     -H "X-M2M-Origin: MyAEID " \
     "https://onem2m.onetransport.io/ONETCSE01/MyAEName/MyContainer"
```

Example response:

```
> GET /ONETCSE01?rcn=6 HTTP/1.1
> Host: onem2m.onetransport.io
> User-Agent: curl/7.47.0
> Accept: application/json
> X-M2M-RI: rcn
> X-M2M-Origin: MyAEID
>
< HTTP/1.1 200 Content
< Content-Type: application/vnd.onem2m-res+json
< X-M2M-RI: rcn
< X-M2M-RSC: 2000
< Content-Length: 158
<
{
  "m2m:cb": {
    "ch": [
      {
        "#text": "/ONETCSE01/ONETCSE01Acp",
        "-nm": "ONETCSE01Acp",
        "-typ": "1"
```

```
      },
      {
        "#text": "/ONETCSE01/MyAEName",
        "-nm": "MyAEName",
        "-typ": "2"
      }
    ],
    "pi": null
  }
}
```