Project Report
Masters of Science in Internetworking

Testing Internet of Things Data Management(IoTDM) Middleware

Submitted by
Kanwaljot Singh

Project Mentor

Mr. Gurpreet Nanda
Senior Solution Architect
Fujitsu

Acknowledgment

I am pleased to acknowledge Mr. Gurpreet Nanda for his invaluable guidance during the course of this project. The knowledge that I gained from the Software Defined Networking course taught by him really helped for the smooth development of this project and without his guidance, this project would have been an uphill task.

Further I want to thanks the University of Alberta for the availability of the Labs and different Tools which were crucial for the project work.

Last but not the least, I am grateful to the members of IoTDM slack group who co-operated with me regarding some issues.

March 2017 Kanwaljot Singh

Abstract

The Internet of Things (IoT) presents a vast potential of technology deployment and business opportunities. Today, IoT products performing similar tasks are densely deployed but providing interoperability between this heterogeneous IoT ecosystem is one of the most important challenge in IoT Technology and Industry. Multiple Platforms are coexisting and competing with each other and they have their own information and data models. Same information from a camera or sensors is described through very different format and values. This makes it difficult for application providers to develop global-wide killer services.

To overcome this problem Internet of Things Data Management(IoTDM) open source project was launched where a common data-centric middleware will be developed. This middleware will be onM2M compliant and applications can be developed to retrieve IoT data uploaded by any device. The oneM2M Global Partnership is developing standards for Machine to Machine communications enabling large scale implementation of the Internet of Things(IoT). IoT data from a specific platform is transformed into the unified data structure & stored in a single data store. In the scope of this project, I will be performing Create Retrieve Update & Delete requests to this Data Store in form of various Tests and these tests are essential for any Application Development.

CONTENTS

1 INTRODUCTION

1.1 Overview

The Internet of Things Data Management (IoTDM) on OpenDaylight(ODL) project is about developing and testing a data-centric middleware that will act as a oneM2M compliant IoT Data Broker and enable authorized applications to Create Retrieve Update & Delete(CRUD) IoT data uploaded by any device. Currently we don't have a common middleware platform in the IoT domain and all the end to end implementations are vendor specific which lead to difficult integration, vendor Lock-in and less competitive market.

The OpenDayLight platform is used to implement the oneM2M data store which models a hierarchical containment tree, where each node in the tree represents a oneM2M resource. IOT devices and applications interact with the resource tree over standard protocols such as CoAP and HTTP. The oneM2M Global Partnership is enabling large-scale implementation of the Internet-of-Things by developing standards for Machine-to-Machine communications. Data-centric paradigm is used to ensure that the network as a distributed IoT platform, provides a single version of the global data space to all the interested applications which is not possible in a message-centric paradigm. Network traffic and application processing is also highly optimizes as devices or applications join and leave the IoT domain.

Initially, the oneM2M resource tree will be used to retrieve data and possible applications or inventory/device management systems or big data analytic systems could be designed later to make sense of the data collected. At some point, applications also need to configure the devices & features or tools will have to be provided to enable configuration of the devices. Applications could be designed to respond to the network conditions, user input, programmable rules or policies possibly triggered by the receipt of data collected from the devices.

1.2 Problem Description

This project is about Internet of Things and we have created a middleware on top of OPENDAYLIGHT which is used as a development platform to build common data repository. The Goal is to produce IOT middleware on OpendayLight based on oneM2M specifications with Basic set of resources supported for minimum implementation which will get it started.

The very first challenge that will be addressed is applications from different IoT platforms cannot understand each other due to lack of common understanding on information and data models. IOT solutions are composed of many different components which are unrelated to each other. Every IOT solution is built for a purpose but these specific solutions can't interact with each other. That's why we have Many devices, Many applications, Many Protocols even for a similar kind of purpose. For example, The security cameras for two different implementations perform similar function & collects videos/data but still may not integrate together as one implementation understand a different data model from the other.

The second challenge which will be addressed is Solution Lock-in with the vendor. Nowadays when we buy an IOT solution, we are actually buying a full IOT system which works end to end which makes the consumer to Lock into the system itself. For example A video surveillance IOT solution will contain cameras , back end system to collect and store information, Video management apps to access the data. What if after some time we are not satisfied with the video quality of the cameras or we what to install some new cameras of a different vendor then how will we going to integrate new cameras into the existing deployment that we have. Most likely we can't do it or we have to deploy a parallel backend system as well. And same is if we are not happy with the backend system then we can't simply change it with a new one from a different vendor and keep our cameras because systems from different vendors are not compatible to each other.

Now another way to deploy this is to have an IOT middleware which sits in the middle. This middleware has the data repository whose standardization is open source and is not vendor specific which allows vendors to develop devices or IOT APPs to interact with the middleware using standard Plug-ins.

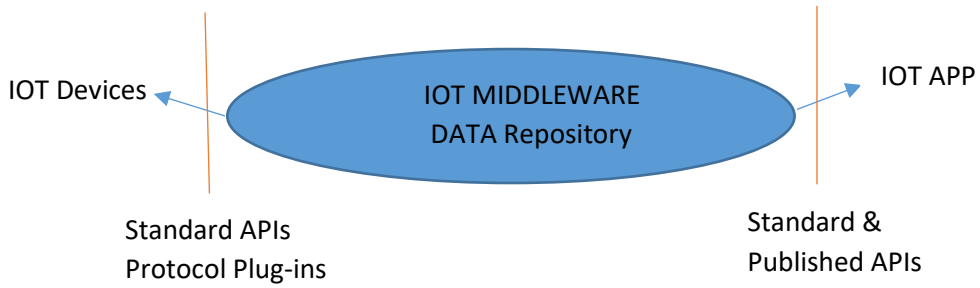


Figure 1

Developing a common IOT middleware will help the consumers to choose and assemble their IOT solutions from different vendors, which will create a more competitive and consumer friendly market.

1.3 Project Objective

Different IoT devices from different vendors are gathering a tons of information, which should be managed, stored and accessed via reliable processes which leads to developing a common IoT middleware so that we can connect any device to any system. An IOT middleware will standardized the APIs towards the devices and towards the application and we will interact with the middleware containing data repository.

In this Report, first I will explain the detailed structure of resource tree of the data repository including different Resources , Containers and their Attributes. After this I will perform various Tests to send Create, Retrieve, Update & Delete requests to the IoT middleware running on OPENDAYLIGHT using Postman Tool. We will see how we can change or create

different resources & how we can update different values of different attributes of these resources. This IoT middleware is still under development and although our APIs can access this Data repository using the simple & extensible REST Technology but work to support various other Northbound and Southbound protocols is still underway. The web based REST mechanism utilizes the HTTP GET, PUT, POST and DELETE commands and access to data involves referencing resources on the middleware.

This Testing will be the key for implementation of Remote Procedure Calls for Create Retrieve Update Delete and notifications for oneM2M complaint IoTDM Systems to build new Applications.

2 TECHNOLOGY & TOOLS

2.1 Internet Of Things(IoT):

Today, The Internet of Things (IoT) presents a huge potential in terms of both, technology deployment and business opportunities. IoT is basically an infrastructure of the information society. IoT allows objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems. This results in improved efficiency, accuracy and economic benefit in addition to reduced human intervention.

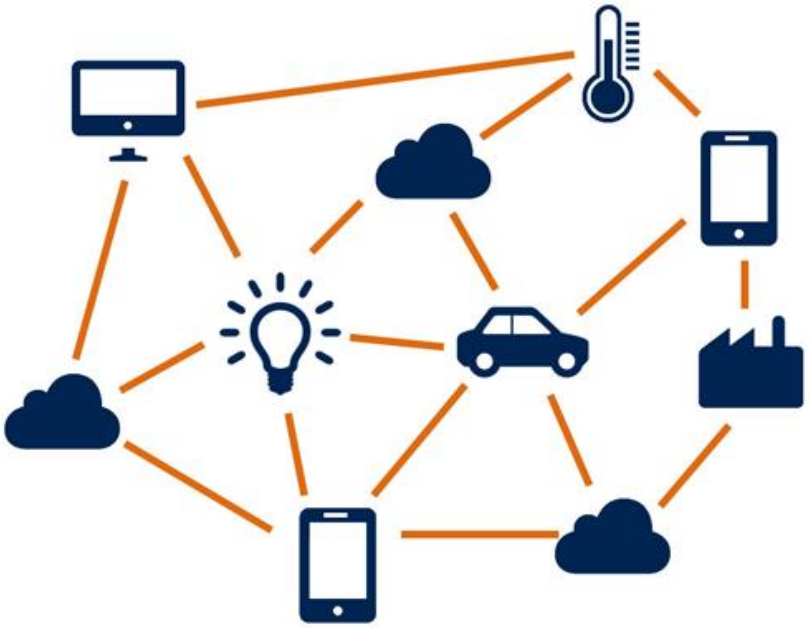


Figure 2 : Internet of Things

If anything has an on and off switch to it then chances are it can be a part of IoT. Everything from phones, cameras, Keys, Cars, various sensors etc are now connected to the network. IoT has a potential to impact not only how we live but also how we work. For Example What if our alarm wakes us up in the morning and then notifies our coffee maker to start brewing coffee for us. What if in a heavy traffic our car notifies the other party that you will be late. Isn't that's more convenient That's why it will definitely change our life.

The reality is that the IoT allows for virtually endless opportunities and connections to take place, many of which we can't even think of or fully understand today. The new rule for the future is Anything that can be connected, will be connected. Therefor It's not hard to see how and why the IoT is such a hot topic today; it certainly opens the door to a lot of opportunities and also to many challenges. Even though the IoT products performing similar tasks are densely deployed but still service providers are deploying multiple dedicated infrastructures per IoT segment. Interoperability between heterogeneous IoT ecosystems is one of the most important challenges in IoT technology and industry today.

2.2 OPENDAYLIGHT

OpenDaylight(ODL) is a highly available, modular, extensible, scalable and multi-protocol controller infrastructure built for SDN deployments on modern heterogeneous multi-vendor networks. OpenDaylight provides a model-driven service abstraction platform that allows users to write apps that easily work across a wide variety of hardware and south-bound protocols. The prime feature of ODL is its ability to support multiple protocols. Prior to the advent of OpenDaylight, almost every general pupose SDN controller used OpeFlow protocol as the sole southbound protocol. OpenDaylight has enjoyed its success due to the support from the Linux Foundation and CISCO.
In this project ODL is used as a platform on which our data centric middleware is developed and various APIs can be developed using northbound plugins of the ODL while our devices will get connection using southbound plug-ins where HTTP, CoAP bindings are used.

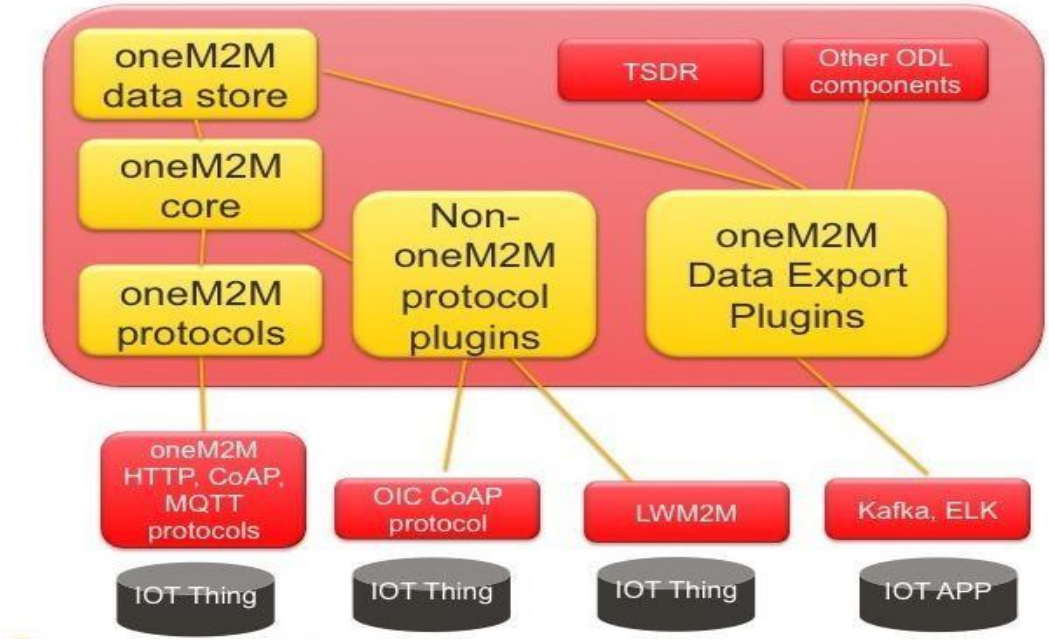


Figure 3 : OpenDaylight

OpenDaylight offers RESTful APIs for creating external applications & RESTful applications require limited software development expertise and are easy to develop. Further OpenDaylight supports various other southbound protocols like BGP, MPLS, NETCONF which makes it a leading preference for creating any business solution.

2.3 POSTMAN TOOL:

POSTMAN is a powerful HTTP client for Testing. Postman makes it easy to test, develop and document APIs by allowing users to quickly put together both simple and complex HTTP requests. Postman has a very clean and intuitive user interface, with most key features accessible within one click. The learning curve for using the program is very low & most users are able to start building and testing API calls very quickly. One big reason for Postman's ease

of use is its automation capabilities, to automate the process of making API requests and testing API responses, allowing developers to establish a very efficient workflow. The response viewer is one of the most important features of the Postman app. API responses are separated in the viewer, with body and headers located in tabs. The status and time codes are displayed adjacent to the tabs. The response viewer also displays the results of API tests, we will see that later in the screenshots of various tests performed.
The Postman REST Client has many other useful functions and features, including keyboard shortcuts, header presets, keyword filter for history and collections, bulk upload/import, and the ability to save API responses to disk. Postman also includes many features designed to dramatically reduce the time needed to test and develop APIs.

2.4 REST Architecture

RESTCONF is a protocol where requests are bundled in easier-to-use REST messages and controller in turn translates these to individual requests with content remaining fundamentally the same. REST uses HTTP or HTTPS and APIs based on it are called RESTful interfaces. This technology has been used primarily for access to information through web service. Web based REST mechanism utilizes the simple HTTP GET, PUT, POST and DELETE commands. Access to data involves referencing resources on the target device using normal and well-understood URL encoding. Requesting entities can access the defined configuration components on a device using REST resources which are represented as separate URLs. It is very straightforward to secure REST communications. Simply by running this web-based protocol through HTTPS adequately addresses security concerns. This has the advantage of easily penetrating firewalls, a characteristics not shared by all network security approaches. RESTCONF is an implementation of Data model using JSON over HTTP.

2.5 oneM2M

oneM2M's architecture and standards are global initiative for Machine to Machine communications which are designed to be applied in many different industries and take account of input and requirements from any sector.

The purpose of oneM2M is to develop technical specifications of a Common embedded IOT middleware across various verticals to minimize standards fragmentation which enables the interoperability across various devices.

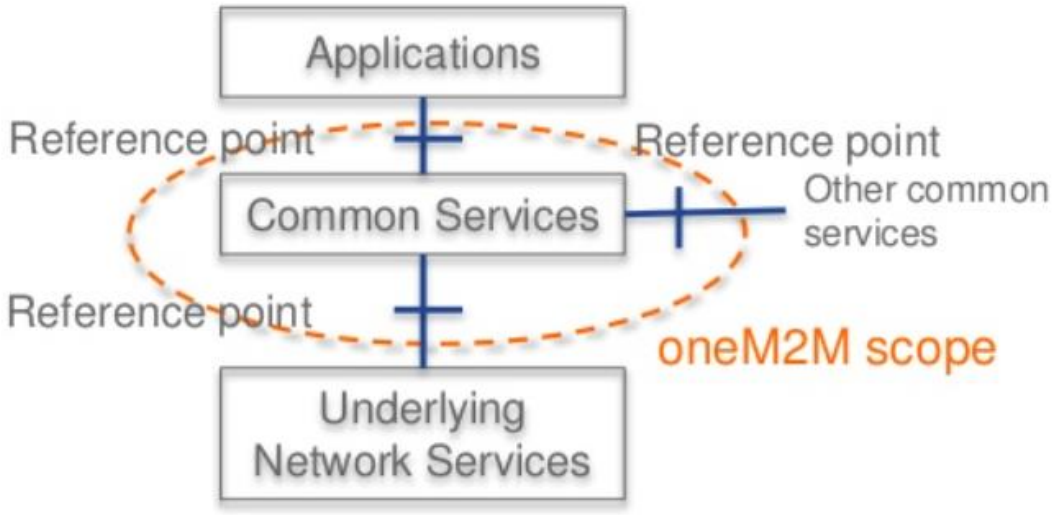


Figure 4 : oneM2M scope

OneM2M provides set of services which will provide data exchange, remote device management, security and access control and connectivity handling. These common services have interfaces to applications, Underlying Network Services and Other common services. Underlying Network provides value added services to the Common Services such as QoS, Device management, Location services and Device triggering etc.

2.6 IOTDM Architecture

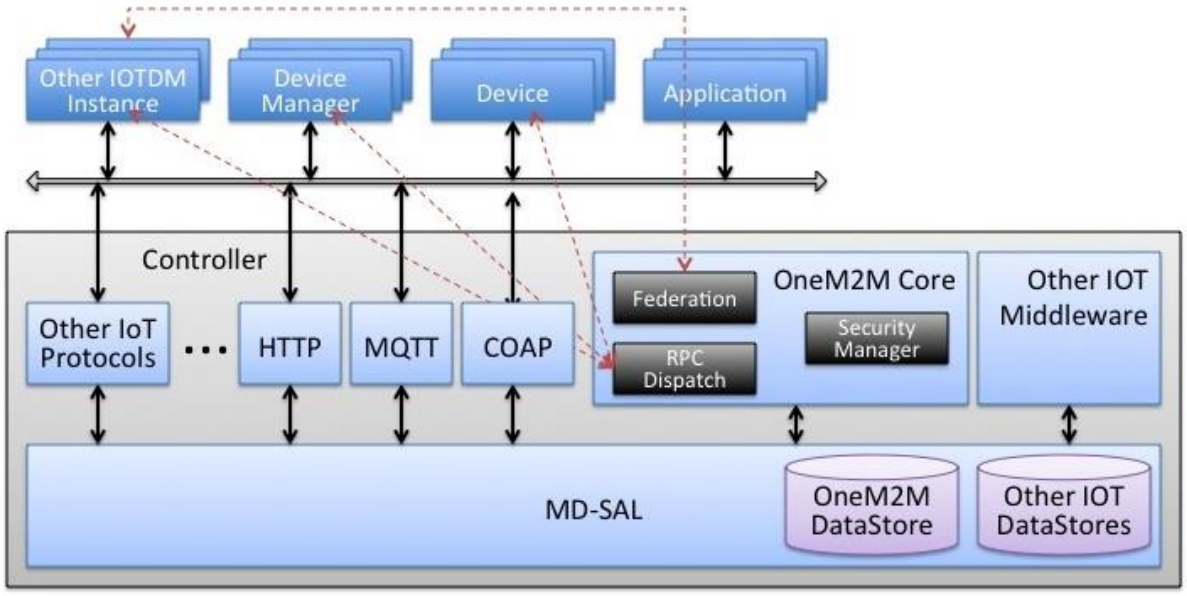


Figure 5 : IOTDM Architecture

In this project of developing data-centric middleware we used infrastructure of OpenDaylight to build our applications. Some plug-in components & various pieces of Service Abstraction Layer of ODL are reused & oneM2M Data collection function(IOTDM) coexist on top of the Model-driven Service Abstraction Layer. The IOTDM application plugins will interact with both data producers (IoT devices such as sensors, and IoT management systems) and data consumers. The interaction with IoT devices is through a variety of IoT Protocol plugins, such as CoAP, HTTP, etc. and the data consumers like APIs interact using REST interfaces.

Some of the rich feature set of this Architecture are as follow.

- Very flexible to support various or custom protocol plugins
- Model-driven Service Abstraction Layer and transactional data store is a natural best fit for the oneM2M resource containment tree.
- Using karaf, Distribution and deployment capabilities are also very flexible.
- Clustering support is provided for scale and performance

3 oneM2M FUNCTIONAL ARCHITECTURE :

oneM2M functional architecture takes Underlying Network-independent view of the end-to-end services and focuses on the Service Layer aspects. The Underlying Network is used for the transport of data.

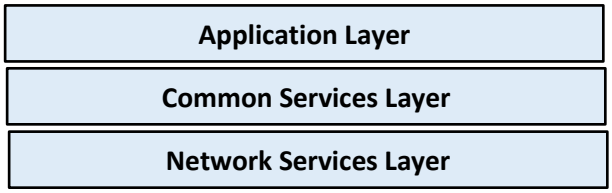


Figure 6: oneM2M Layered Model

All IOT entities are represented as Resources in a tree:

- Applications, Devices, Data, Groups, Access Rights, Billing Policies etc.

Attributes of the resources describe how the system manages the resources

- Time to live, creation time, labels, different IDs etc.

As our data from our IOT devices can grow in an exponential rate which in turn will grow our resource tree therefore we need to keep it in check. This could be done using attributes. For example we can say I want to collect data for next one hour only, or collect the readings from a device after every 5 minutes only etc.

The tree representation is standardized not its implementation therefore by understanding a common tree structure, IoT components can interoperate. If everybody know how to navigate through the tree they will know how to find information

3.1.1 RESOURCES:

In the oneM2M system all entities are represented as resources such as AE's, CSE's, data. Addressing of these resources are unique and as a representation of such resources, a resource structure is specified. Procedures for accessing such resources are also specified.

Resources can be of three catagories:

- Normanl Resources : The complete set of representations of data is done in Normal Resources which constitutes the base of the information to be managed.
- Virtual Resources: To trigger processing or to retrieve results, Virtual Resources or Virtual attributes are used but these resources do not have a permanent representation in a CSE.
- Announced Resources: A resource at a remote CSE which is linked to the original resource that has been announced is an announced resource. This also keep some of the characteristics of the original resource

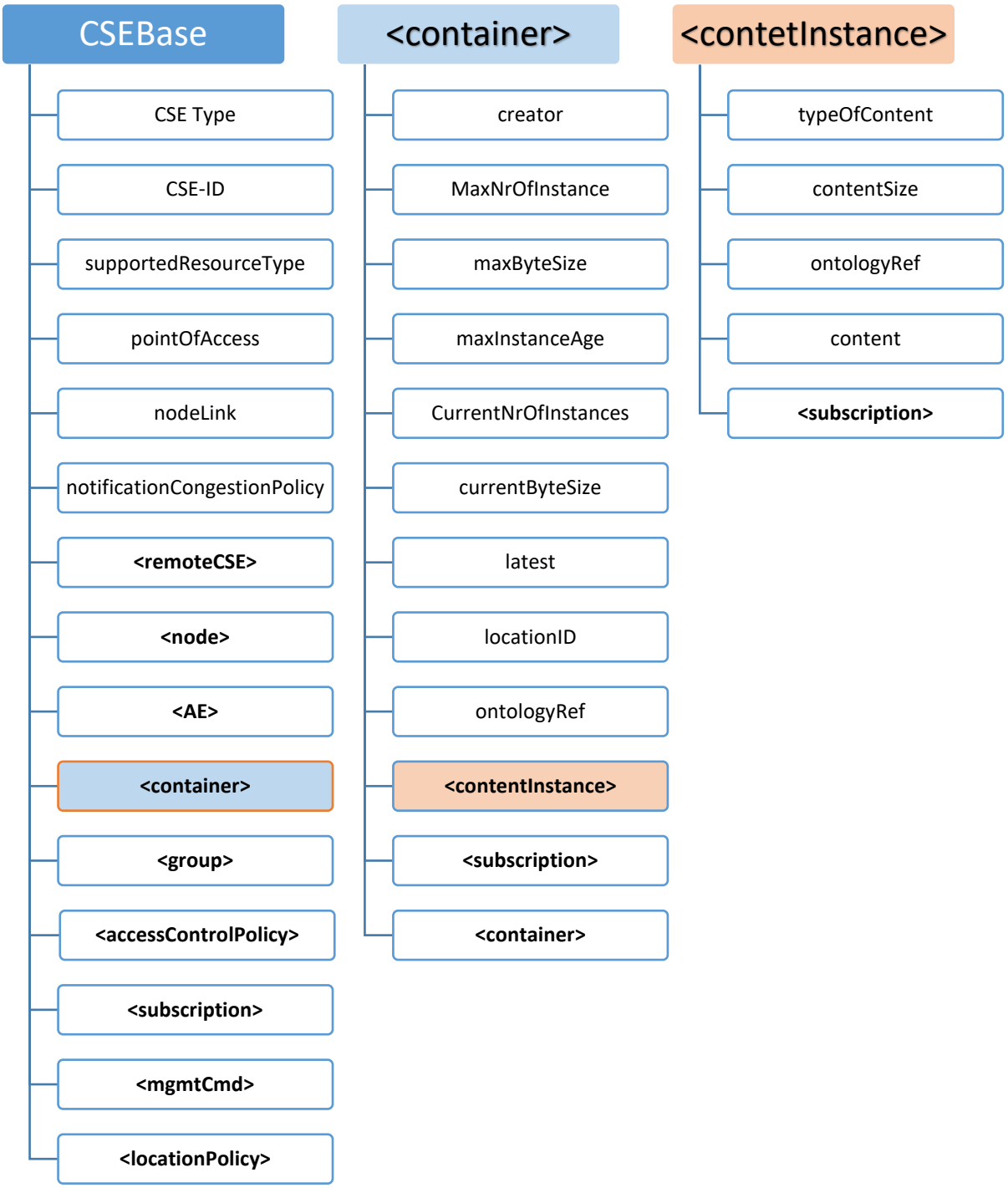


Figure 7: Resources and Attributes

Application Entity(ae): Application Entity is an entity in the application layer that implements an M2M application service logic. Each application service logic can be resident in a number of M2M nodes and/or more than once on a single M2M node. Each execution instance of an application service logic is termed an "Application Entity" (AE) and is identified with a unique AE-ID(aei) assigned by system, App-ID(api) which is mandatory and an optional App Name(apn) attribute.

Common Service Entity(cse): A Common Services Entity represents an instantiation of a set of "common service functions" of the M2M environments. Such service functions are exposed to other entities. Each Common Service Entity is identified with a unique CSE-ID(csi). One of the purposes of CSEs is to enable AEs to exchange data with each othe**r.**

Network Service Entity(NSE): A Network Services Entity provides services from the underlying network to the CSEs

Node(nod):
These are logical entities that are individually identifiable in the M2M System. As Logical Objects, Nodes may or may not be mapped to the physical objects.

Common Services Functions(CSF):
Services provided by the Common Services Layer in the M2M System resides within a CSE and are referred to as Common Services Functions (CSFs). The CSFs provide services to the AEs and to other CSEs and also interact with the NSE. An instantiation of a CSE in a Node comprises a subset of the CSFs. The CSFs contained inside the CSE can interact with each other.

Data Management and Repository (DMR) CSF is responsible for providing data storage and mediation functions. It includes the capability of collecting data for the purpose of aggregating large amounts of data, converting this data into a specified format, and storing it. The data can be either raw data transparently retrieved from an M2M Device or processed data which is calculated or aggregated by M2M entities. The DMR CSF provides the capability to store data such as Application data, subscriber information, location information, device information, semantic information, communication status, access permission

Other Common Service Functions are Discovery, Registration, Security, Subscription & Notification, Location, Group Management etc.

AccessControlPolicy(acp):
Stores a representation of privileges. It is associated with resources that shall be accessible to entities external to the Hosting CSE. It controls "who" is allowed to do "what" and the context in which it can be used for accessing resources

Container(con):
Shares data instances among entities. Used as a mediator that buffers data exchanged between AEs and/or CSEs. The exchange of data between AEs (e.g. an AE on a Node in a field domain and the peer-AE on the infrastructure domain) is abstracted from the need to set up direct connections and allows for scenarios where both entities in the exchange

do not have the same reachability schedule. ContentInstance(cin) represents a data instance in the container resource.

CSEBase(cb):
The structural root for all the resources that are residing on a CSE. Stores information about the CSE itself

Group(grp):
Stores information about resources of the same type that need to be addressed as a Group. Operations addressed to a Group resource shall be executed in a bulk mode for all members belonging to the Group.

RemoteCSE:
Represents a remote CSE for which there has been a registration procedure with the registrar CSE identified by the CSEBase resource.

Subscription(subs):
Subscription resource represents the subscription information related to a resource. Such a resource shall be a child resource for the subscribe-to resource

3.1.2 Resource Attributes

Resource Attributes stores information pertaining to the resource. An attribute has a name and a value. Only one attribute with a given name can belong to a given resource. There are some common as well as special Attributes listed below:

SupportedResourceType(srt): This Read Only (assigned at creation time. and then cannot be changed) attribute identifies the supported type of the resource. Each resource shall have a resourceType attribute

ResourceID(ri): This attribute is an identifier for the resource and shall be provided by the Hosting CSE when it accepts a resource creation procedure. The Hosting CSE shall assign a resourceID which is unique in that CSE.

ResourceName(rn): This attribute is the name for the resource that is used and may be provided by the resource creator. The Hosting CSE shall use a provided resourceName as long as it does not already exist among child resources of the targeted parent resource. If the resourceName already exists, the Hosting CSE shall reject the request and return an error to the Originator. The Hosting CSE shall assign a resourceName if one is not provided by the resource creator.

ParentID(pi): This attribute is the resourceID of the parent of this resource. The value of this attribute shall be NULL for the CSEBase resource type

CreationTime(ct): Time/date of creation of the resource. This attribute is mandatory for all resources and the value is assigned by the system at the time when the resource is locally created. Such an attribute cannot be changed.

LastModifiedTime(lt): Last modification time/date of the resource. The lastModifiedTime value is set by the Hosting CSE when the resource is created,and the lastModifiedTime value is updated when the resource is updated.

ExpirationTime(et): Time/date after which the resource will be deleted by the Hosting CSE. This attribute can be provided by the Originator, and in such a case it will be regarded as a hint to the Hosting CSE on the lifetime of the resource. The Hosting CSE can however decide on the real expirationTime. If the Hosting CSE decides to change the expirationTime attribute value, this is communicated back to the Originator.

AccessControlPolicyIDs(acpi): This attribute contains a list of identifiers of an accessControlPolicy resource basically it's a Pointer which points to acp.

AccessControlOriginators(acor): It's a mandatory parameter which represents the set of Originators that shall be allowed using this access control rule.

AccessControlContexts(acco): It's an optional parameter in an access-control-rule-tuple that contains a list, where each element of the list, when present, represents a context that is permitted using this access control rule.

AccessControlOperations(acop): It's a mandatory parameter in an access-control-rule-tuple that represents the set of operations that are authorized using this access control rule.

Labels(lbl): A list of tolens used as keys for discovering resources.

PointOfAccess(poa): The list of addresses for communicating with the registered Application Entity provided by Underlying Network e.g. IP address, URI etc. This attribute shall be accessible only by the AE and the Hosting CSE.

RequestReachability(rr): If the CSE that created remoteCSE resource can receive a request from other AE/CSE(s) then this attribute is set to "TRUE" otherwise "FALSE.

OntologyRef(or): A reference (URI) of the ontology used to represent the information that is stored in the contentInstances resources of the container resource. If this attribute is not present, the contentInstance resource inherits the ontologyRef from the parent.

AnnounceTo: This attribute may be included in a CREATE or UPDATE Request in which case it contains a list of addresses/CSE-IDs where the resource is to be announced

AnnouncedAttribute: This attributes shall only be present at the original resource if some Optional Announced (OA) type attributes have been announced to other CSEs.

3.1.3 Communication Flows:

Procedures involving CSEs and AEs are driven by the exchange of messages according to the message flows. Depending on the message operation, procedures may manipulate information in a standardized resource structure. The general flow that governs the information exchange within a procedure is based on the use of Request and Response messages either between an AE and a CSE or among CSE's. Requests from Originator to a Receiver contains To, From, Operation, Request Identifier and other Operation dependent Parameters.

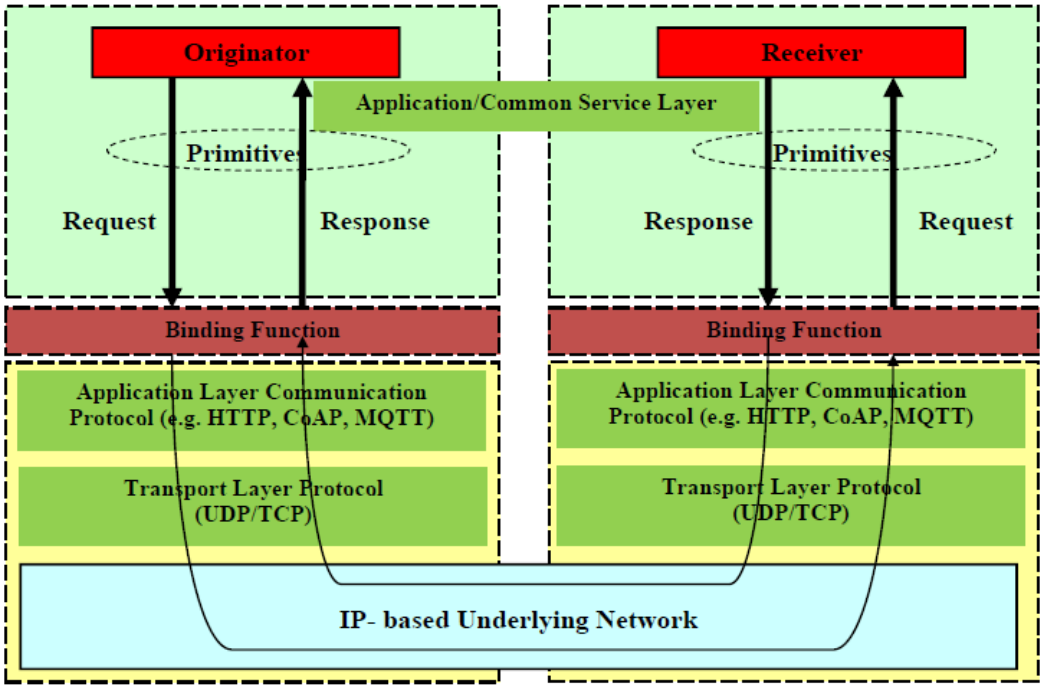


Figure 8 : Communication Flows

Resources has a representation that shall be transferred and manipulated with the verbs identified as Operations which are as follow:

Create(C): This operation shall be used by an Originator CSE or AE to create a resource on a Receiver CSE , also called the Hosting CSE.

Retrieve(R): This operation shall be used for retrieving the information stored for any of the attributes for a resource at the Receiver CSE. The Originator CSE or AE may request to retrieve a specific attribute by including the name of such attribute in the Content parameter in the request message.

Update(U): This operation shall be used for updating the information stored for any of the attributes at a target resource. Especially important is the expirationTime, since a failure in refreshing this attribute may result in the deletion of the resource. The Originator CSE or AE can request to update, create or delete specific attribute(s) at the target resource

by including the name of such attribute(s) and its values in the Content parameter of the request message

Delete(D): This operation shall be used by an Originator CSE or AE to delete a resource on a Receiver CSE ,also called the Hosting CSE.

Notify(N): This operation shall be used for notifying information to be sent to the Receiver.

3.1.4 Resource Addressing & Structure:

An address of a resource is a string of characters used to uniquely identify the targeted resource within the scope of a request to access the resources. The resources in the onM2M system are linked with each other and they respect the containment relationships.

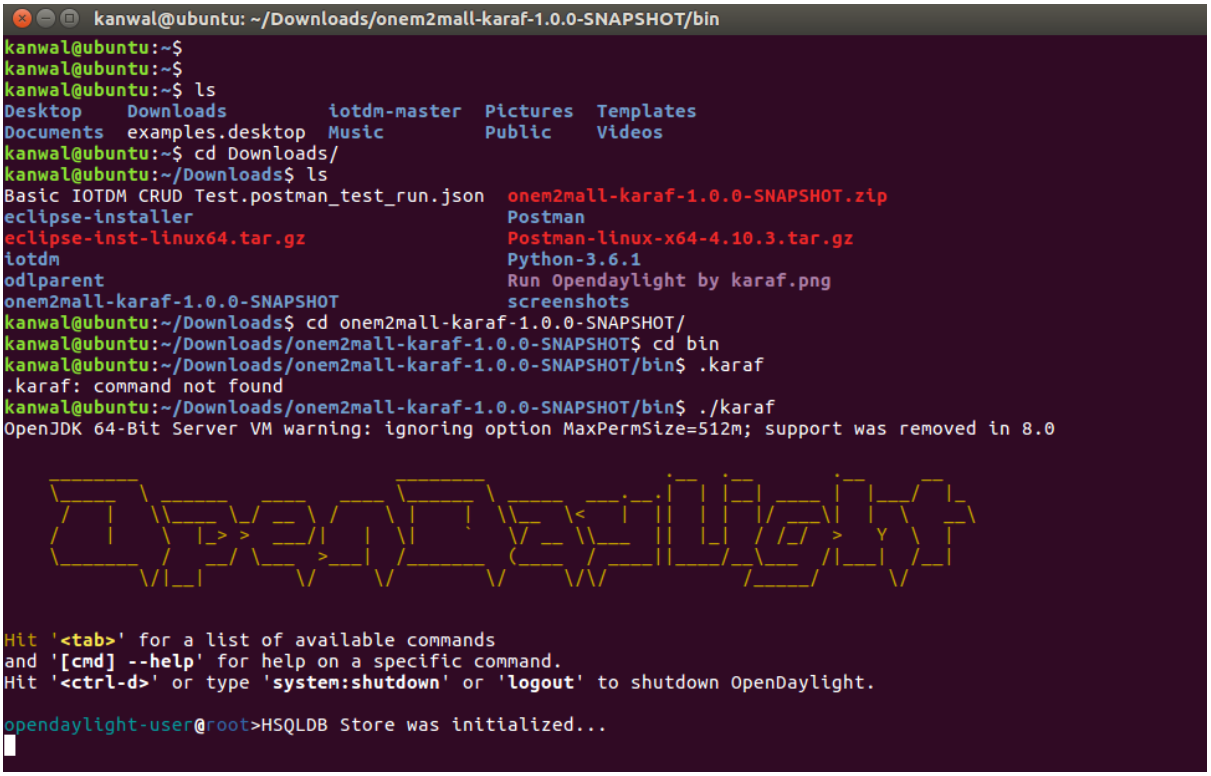
Linked Resource Type	**Linking Resource Type**	**Linking Method (Attribute named)**
AccessControlPolicy	Several (e.g node, AE, remoteCSE, container)	AccessControlPolicyIDs
Node	CSEBase, remoteCSE,AE	nodeLink
CSEBase or remoteCSE	Node	Parent resource of type CSEBase
contentInstance	Contentinstance	contenetRef

Figure 9

4 TESTING

4.1 System Setup

1. Installed Ubuntu, a UNIX based Operating System in the Virtual Machine.
2. Download zip file of IOTDM middleware.
3. Unzip and open bin folder in the terminal
4. Run IOTDM on OpenDaylight
   Command: ./karaf



5. Download and Install Postman Tool
6. Now we are ready to send REST messages to Create, Retrieve, Delete & Update resources on the IOTDM middleware running on the localhost and we can observe the behavior in the response window of Postman.

4.2 Test results:

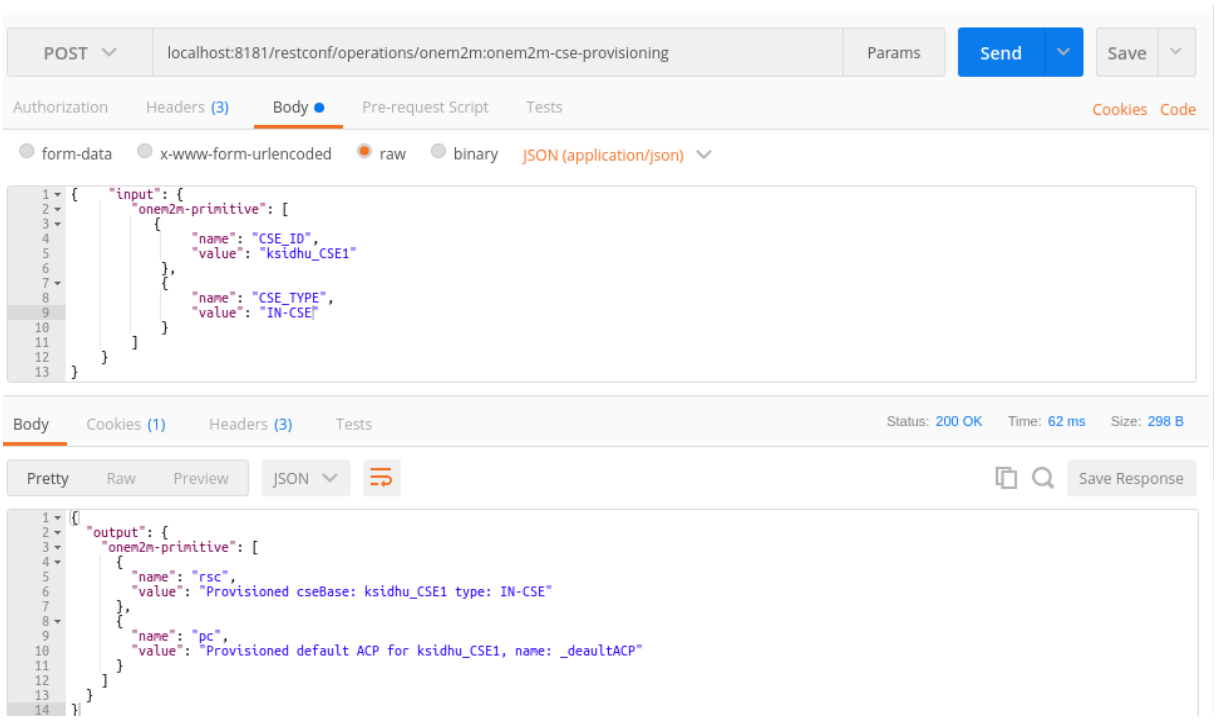
In each test, resource operation comprises a pair of Request & Response primitives which are represented as JSON texts. This JSON text can be stored or exchanged between network entities by following the principle of RESTful architecture.

Test 0:

The very first thing to do is the provisioning of CSE resource. This will be done by sending Post message on RESTful port 8181 as it's a RESTCONF process. In the Body of this request, I specified the CSE_ID as **ksidhu_CSE1** of type **IN-CSE**.

Response:

In the response window we could see the Provisioned cseBase : ksidhu_CSE1 with type:In-CSE. At the same time we got a default Access Control Policy named _defaultACP which could be retrieved or Updated.



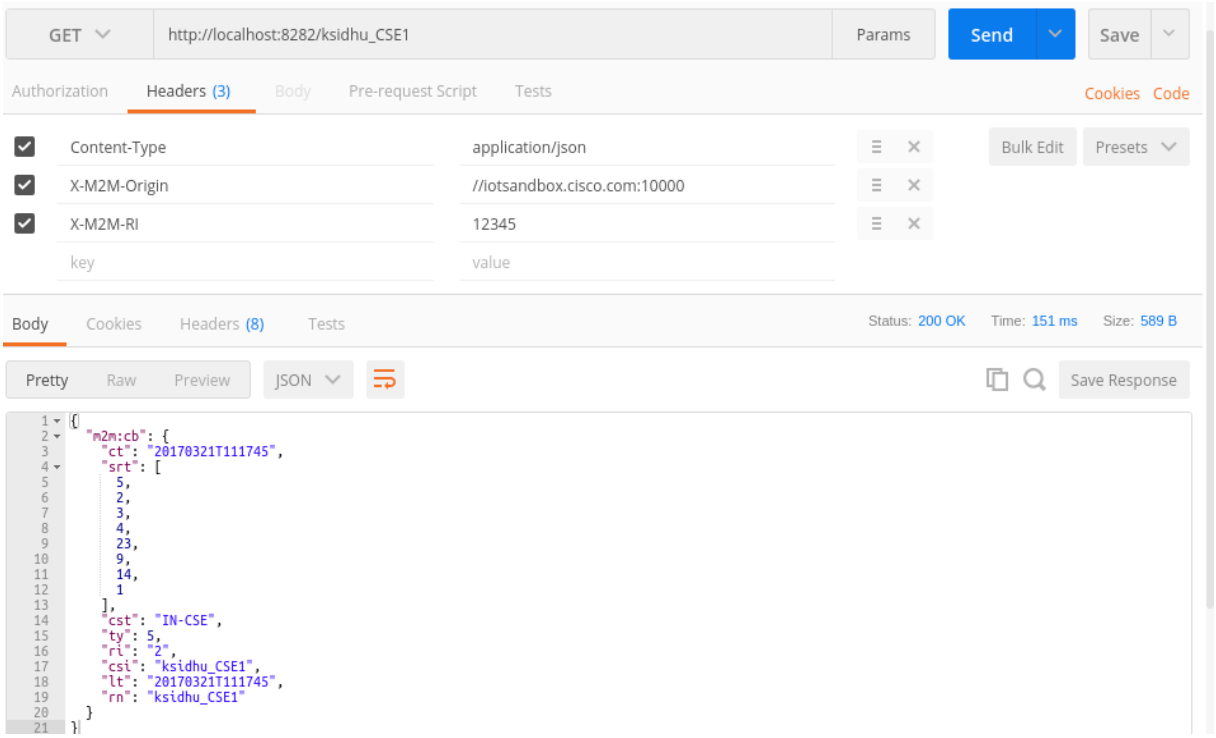
Test 1:

After we provisioned a CSE we can retrieve it via HTTP GET message port 8282.

GET http://localhost:8282/ksidhu_CSE1

Response:

In the response window we could see our retrieved CSE with CSE-ID(csi): ksidhu_CSE1 along with attributes: Creation Time (ct), Last Modified Time(lt), ResourceName(rn), ResourceID(ri) Supported Resource Types(srt) values which corresponds to AE, Group, Node, Container, ResourceContentInstance, Resource Subscription & Access Control Policy.



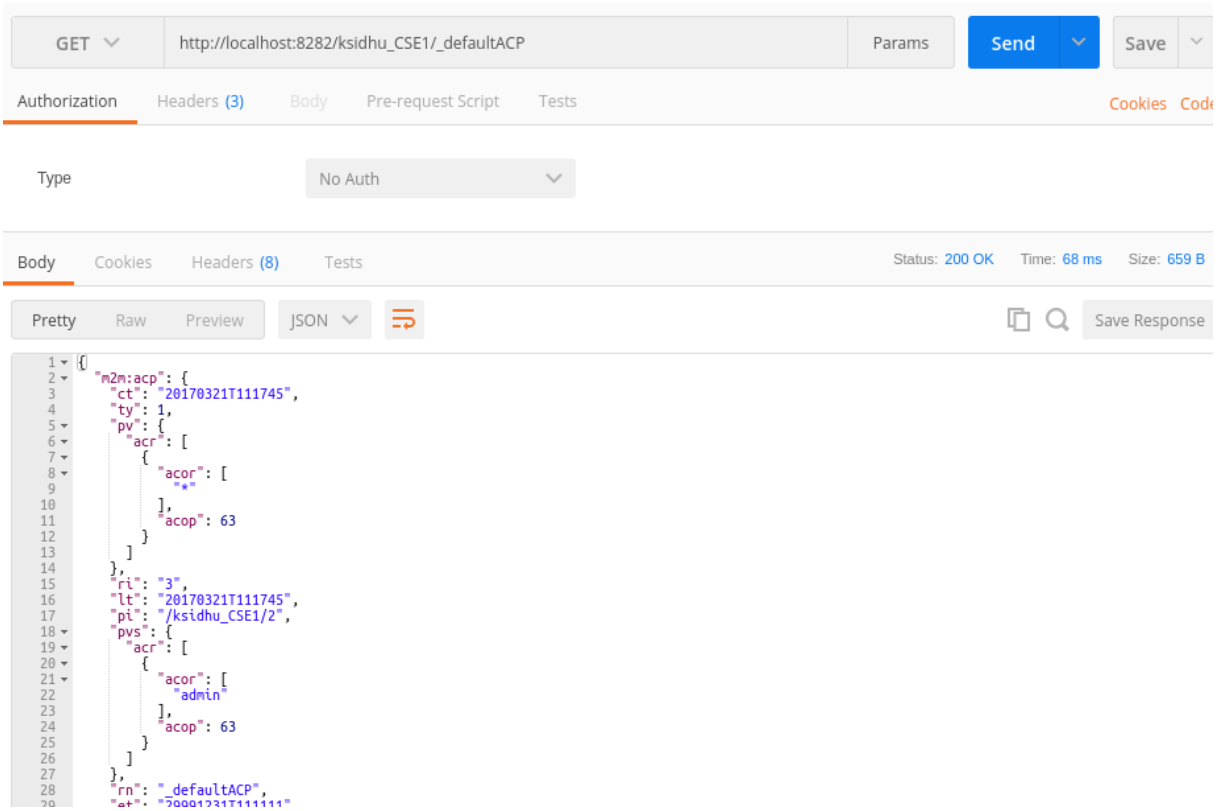
TEST 2:

We can retrieve the default Access Control Policy in ksidhu_CSE1.

GET http://localhost:8282/ksidhu_CSE1/_defaultACP

RESPONSE:

In the response window we could see ResourceName(rn) as _defaultACP with ParentID(pi) as ksidhu_CSE1. “acor” (accessControlOrigin) as ”*” & acop(accesscontrolOperation)”63” indicates that this policy allows anyone to do any operations. We can change this ACP and set the origin part to any user like admin with any operations in its defaultACP.



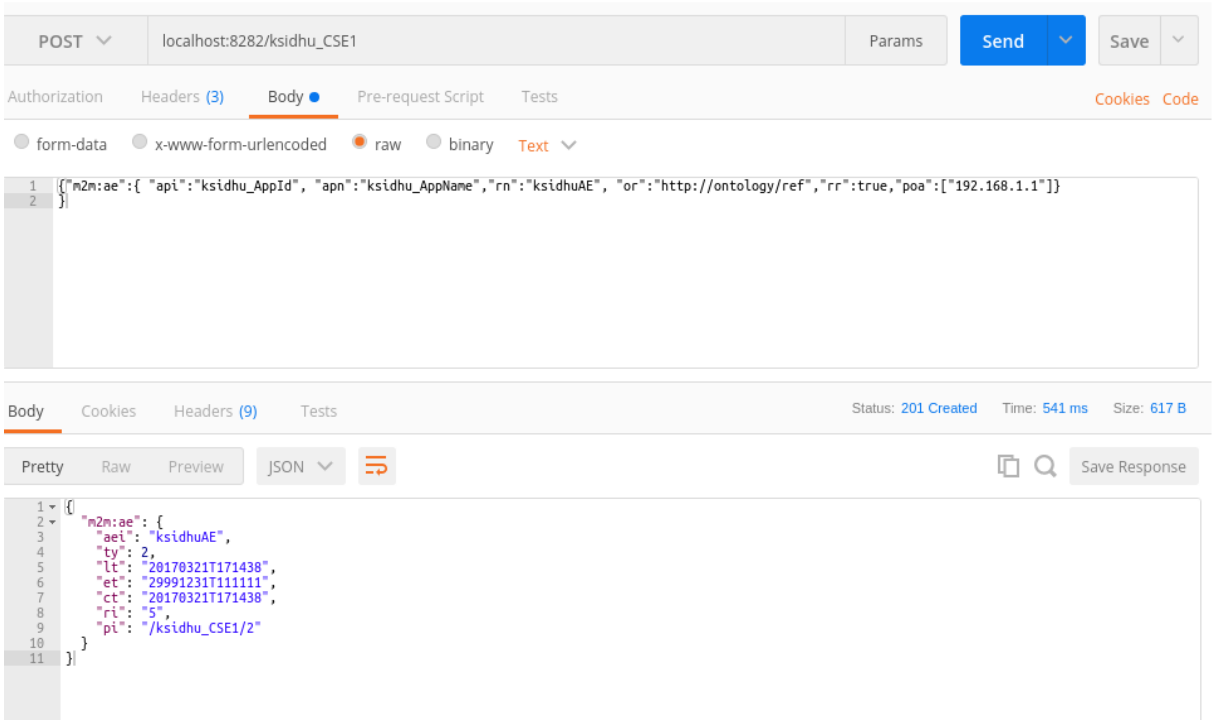
TEST 3:

Register AE to its registrar ksidhu_CSE1 with AppID as ksidhu_Appid, App name as ksidhu_AppName & resource name as ksidhuAE.

POST http://localhost:8282/ksidhu_CSE1

RESPONSE:

After sending this POST message we could see in the response that AEID(aei) ksidhuAE has been created in parentID(pi) ksidhu_CSE1 and the alternate location of the returned data is /ksidhu_CSE1/ksidhuAE.



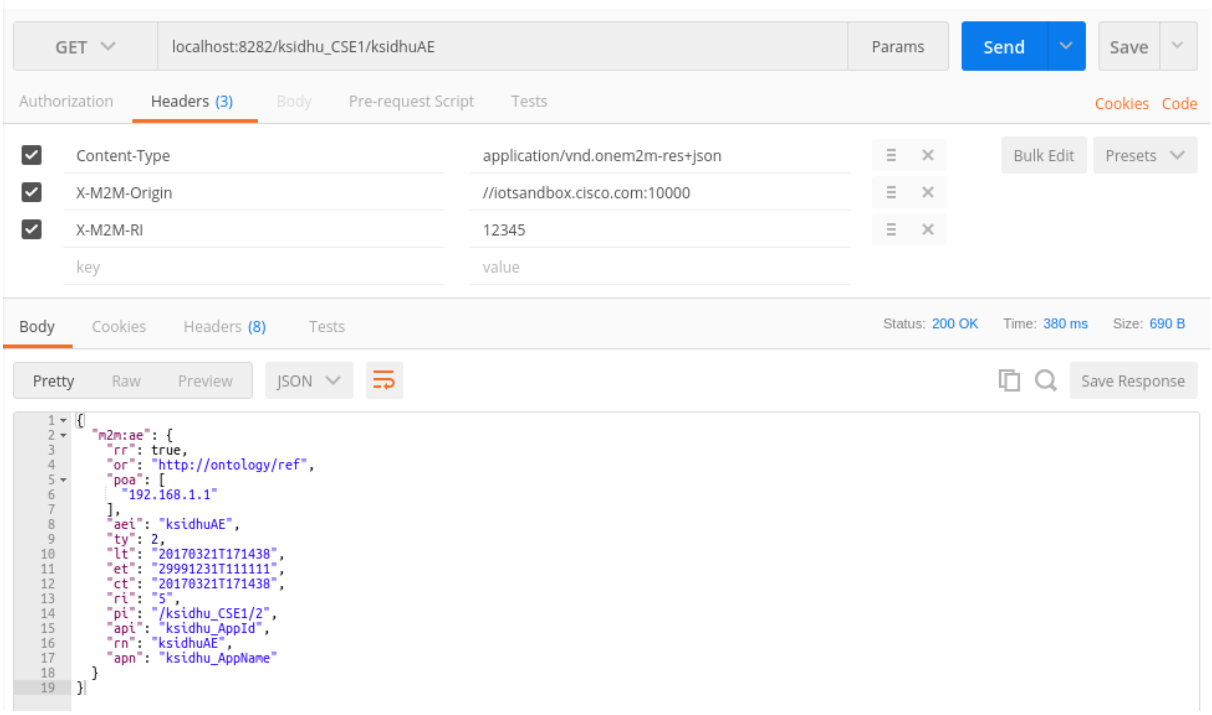
TEST 4:

Retrieve recently created AE from /ksidhu_CSE1/ksidhuAE

GET localhost:8282/ksidhu_CSE1/ksidhuAE

RESPONSE:

In the response window we could see the retrieved ApplicationEntity AEID(aei) ksidhuAE with attributes RequestReachability set to True, PointOfAccess parameter of underlying network as 192.168.1.1, an arbitrary ontologyRef(or) value to represent the information stored.



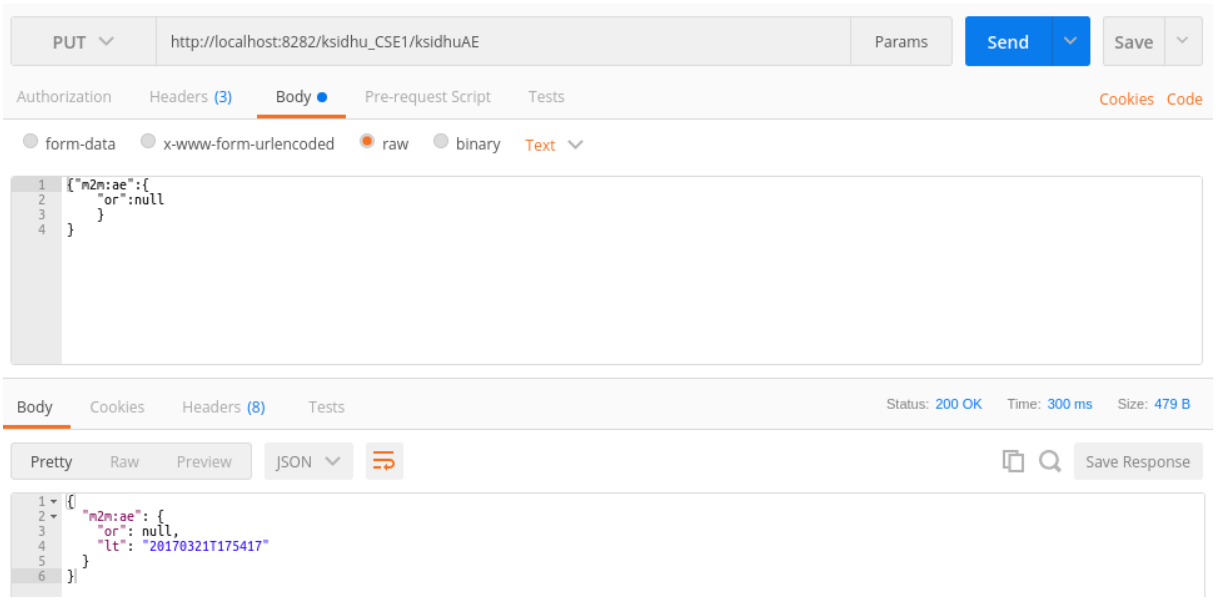
TEST 5:

Update & then Retrieve attribute ontologyRef in AE: ksidhuAE

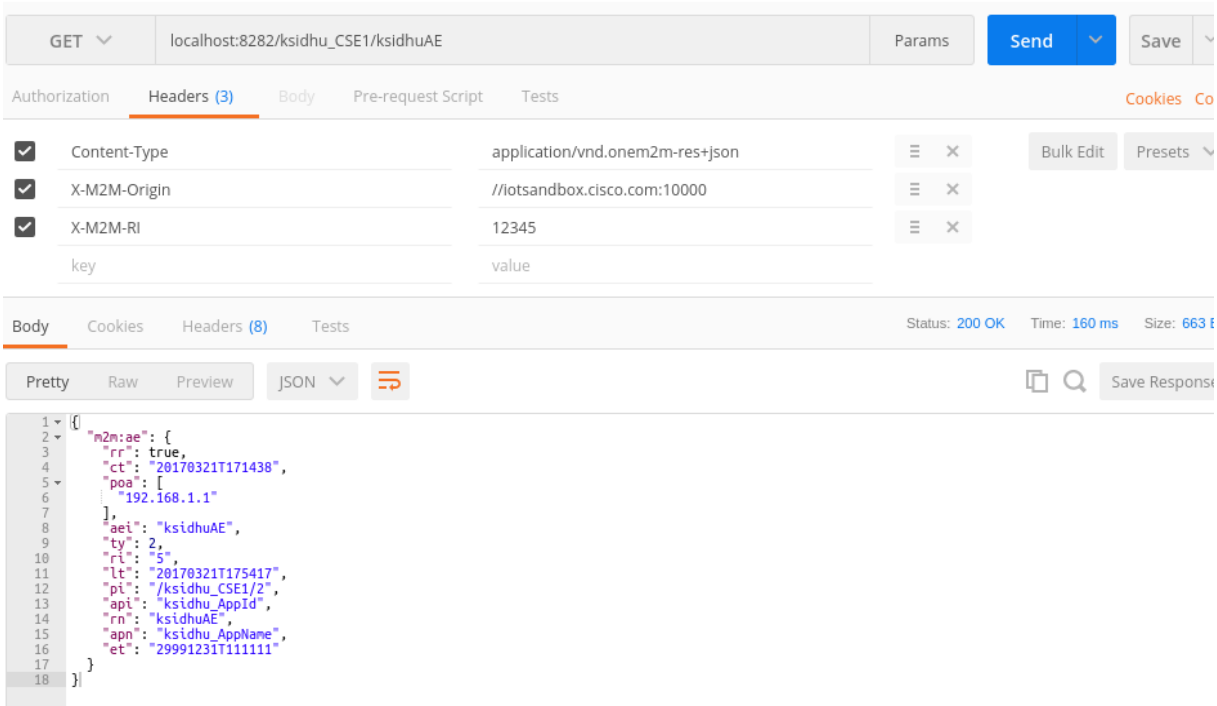
PUT http://localhost:8282/ksidhu_CSE1/ksidhuAE with “or”:null in the Body

RESPONSE:

In the response window of Retrieve we could see that there is no “or” attribute . Therefore, update has been done successfully



Retrieve after AE update



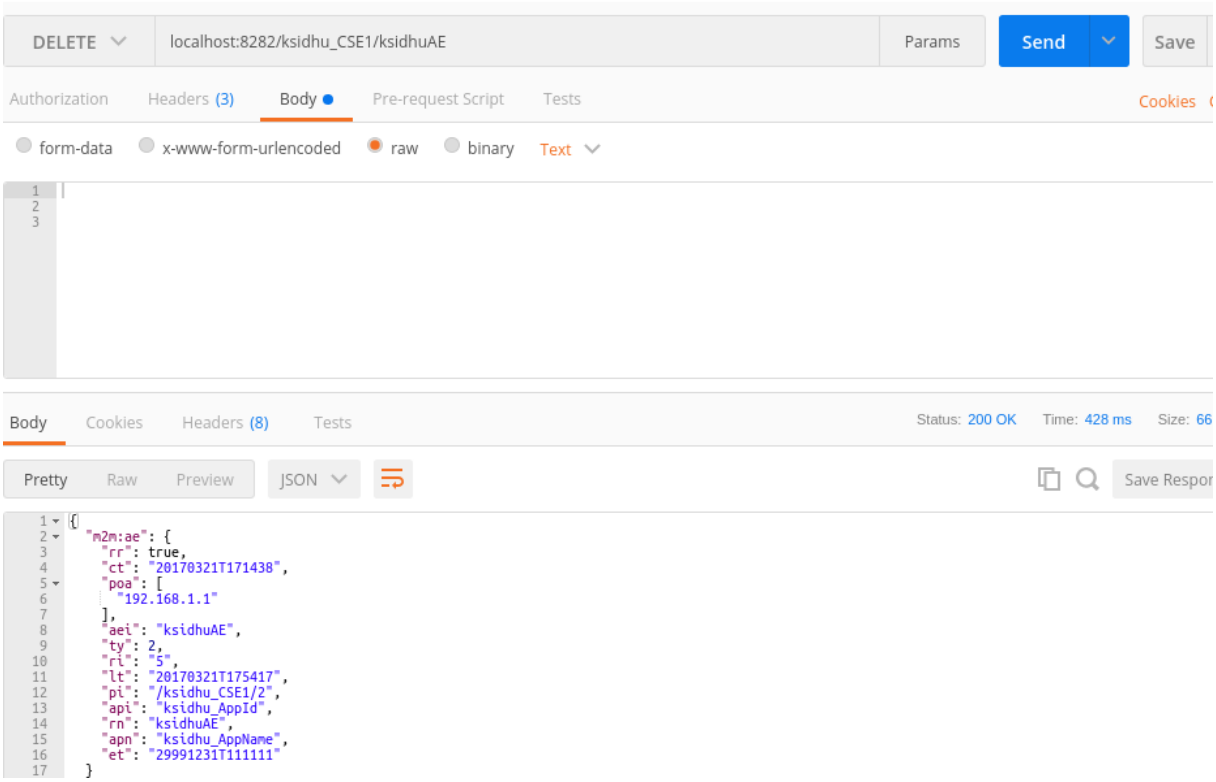
TEST 6:

Update & then Retrieve attribute ontologyRef in AE: ksidhuAE at ksidhu_CSE1

DELETE http://localhost:8282/ksidhu_CSE1/ksidhuAE

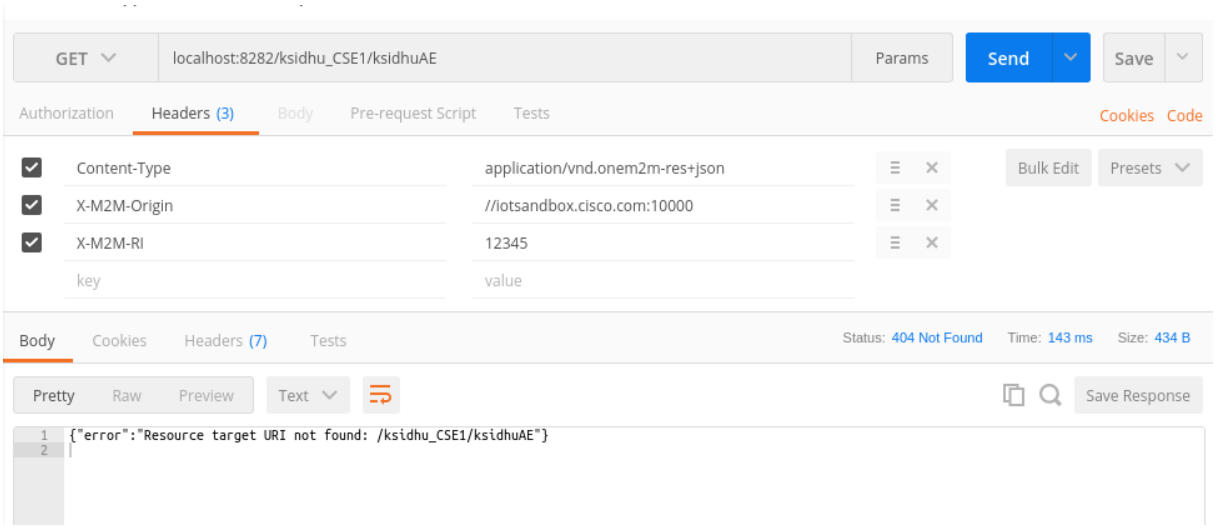
RESPONSE:

In the response window of Retrieve we could see resource target not found as ksidhuAE has been deleted successfully. Also after deteting ksidhuAE once we can not delete that again.



Retrieve after Delete:

GET http://localhost:8282/ksidhu_CSE1/ksidhuAE



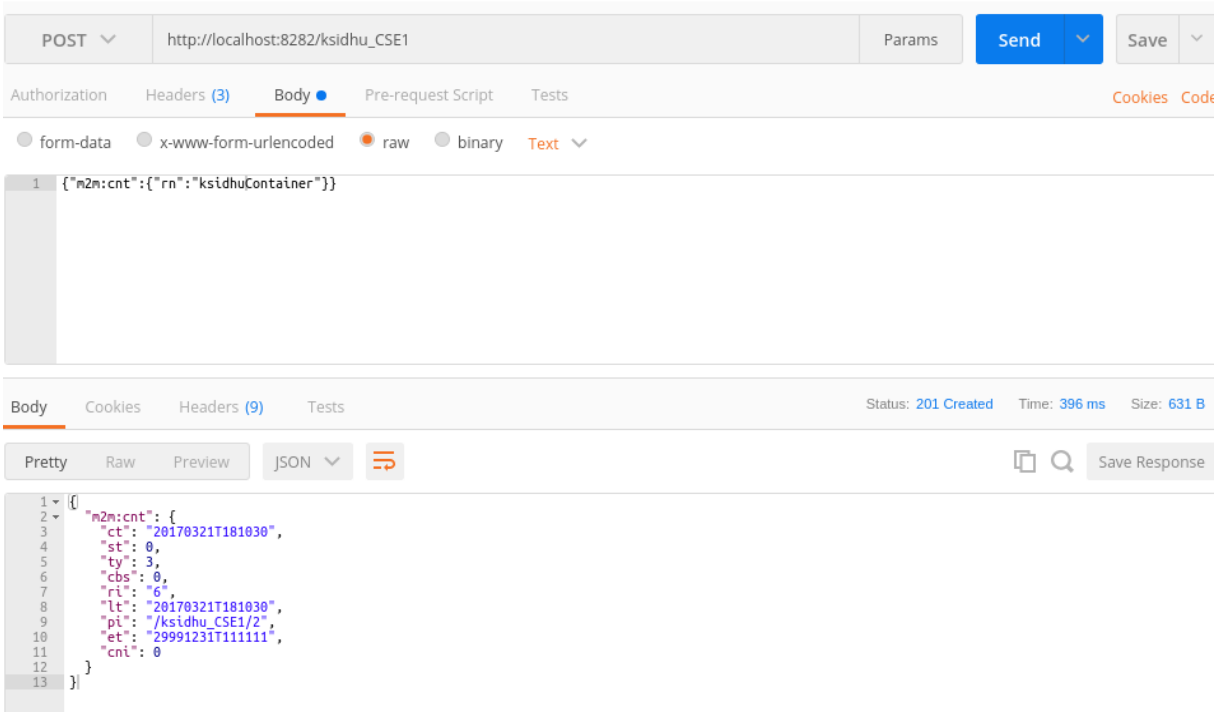
TEST 7:

Create a container: ksidhucontainer in the ksidhu_CSE1.

POST http://localhost:8282/ksidhu_CSE1

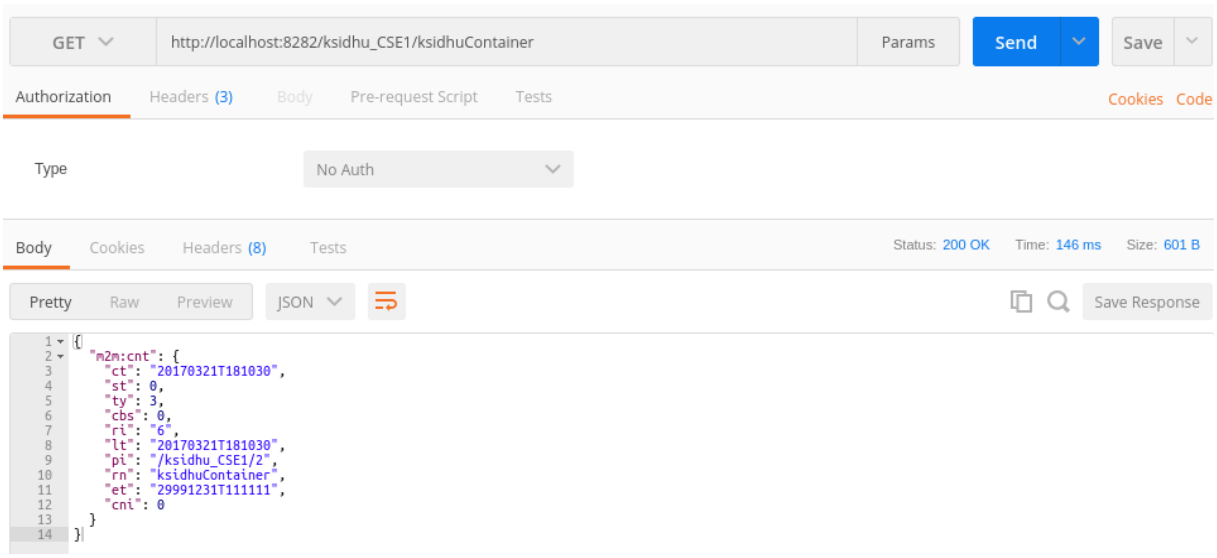
RESPONSE:

ksidhuContainer has been created successfully which we received with GET message. Various similar attributes are also attached to this like parentID, creation time and last modified time as well.



Retrieve Container: ksidhucontainer

GET http://localhost:8282/ksidhu_CSE1/ksidhuContainer



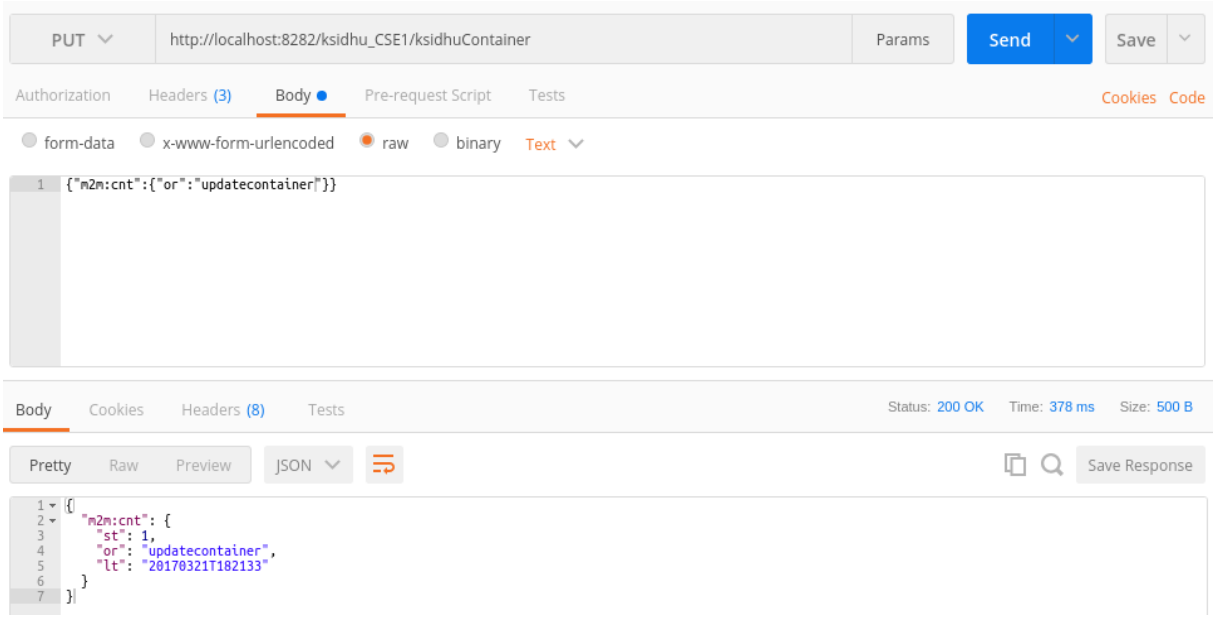
TEST 8:

Update attribute OntologyRef for container: ksidhucontainer in the ksidhu_CSE1.

PUT http://localhost:8282/ksidhu_CSE1/ksidhuContainer

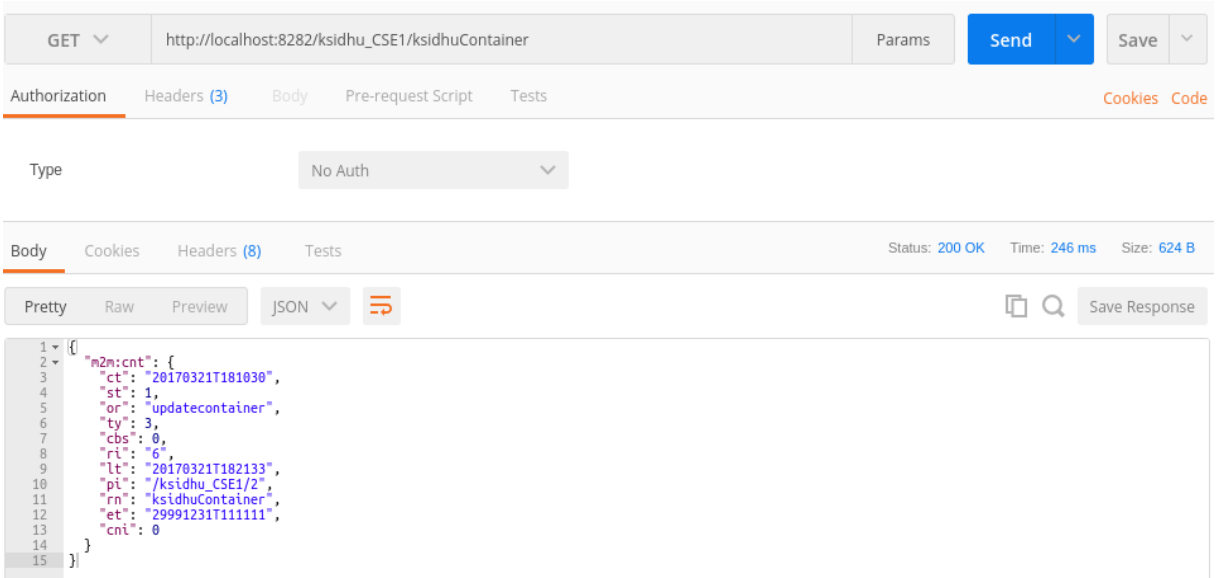
RESPONSE:

ksidhuContainer has been updated successfully and we can see the “or” attribute in the retrieve request.



Retrieve after update:

GET http://localhost:8282/ksidhu_CSE1/ksidhuContainer



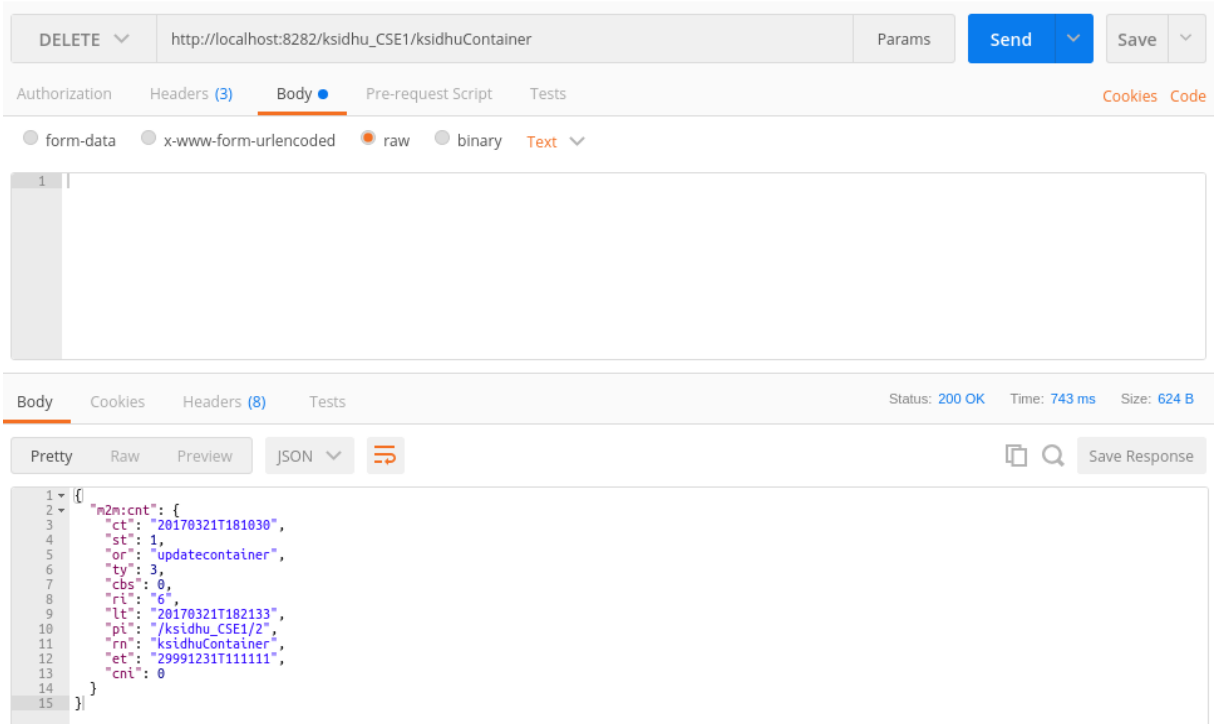
TEST 9:

Delete container: ksidhucontainer in the ksidhu_CSE1.

DELETE http://localhost:8282/ksidhu_CSE1/ksidhuContainer

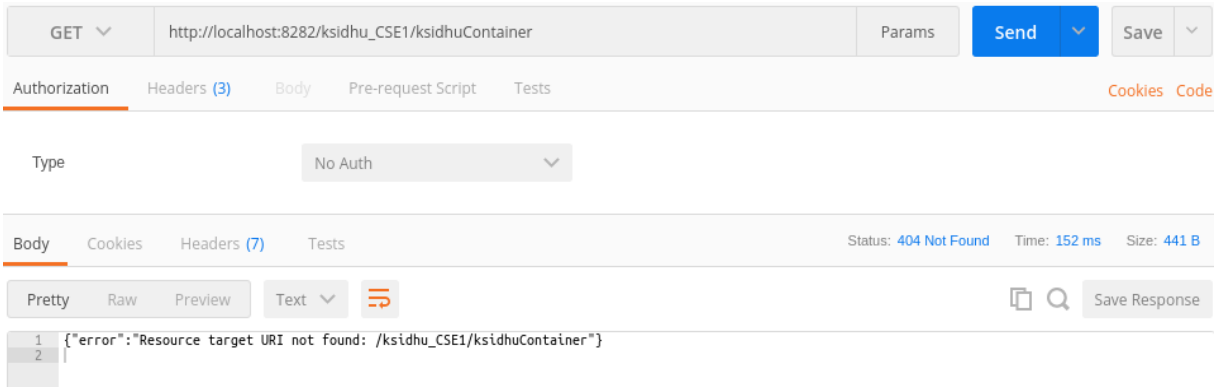
RESPONSE:

ksidhuContainer has been deleted successfully and we can see that target URI not found in the retrieve request.



Retrieve after deleting container

GET http://localhost:8282/ksidhu_CSE1/ksidhuContainer



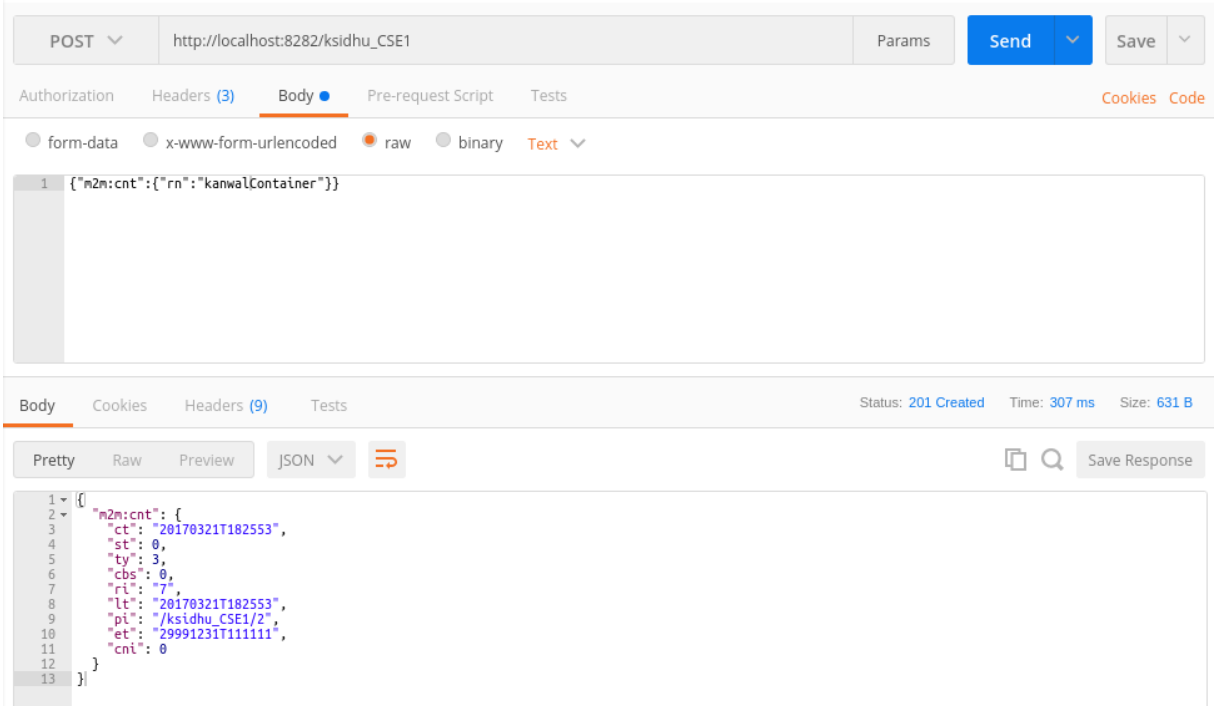
TEST 10:

Create a new container: kanwalContainer in the ksidhu_CSE1 for ContentInstance

POST http://localhost:8282/ksidhu_CSE1

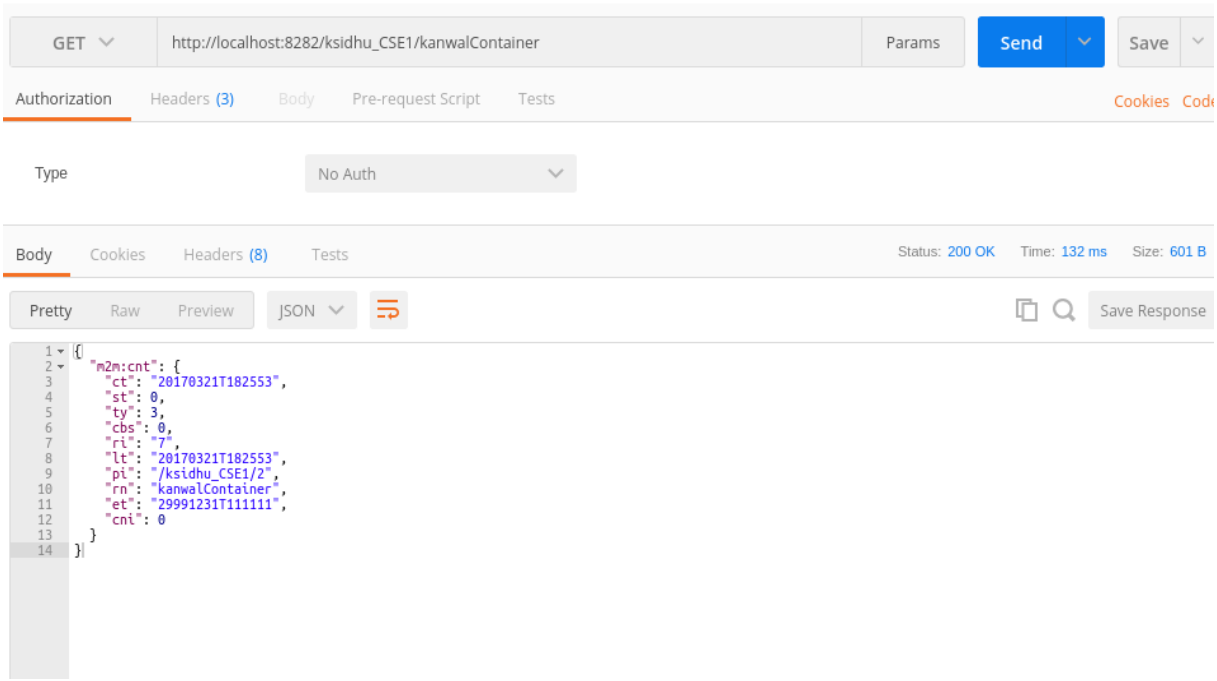
RESPONSE:

kanwalContainer has been created successfully which we received with GET message. Various similar attributes are also attached to this like parentID, creation time and last modified time as well.



Retrieve Container: kanwalcontainer

GET http://localhost:8282/ksidhu_CSE1/kanwalContainer



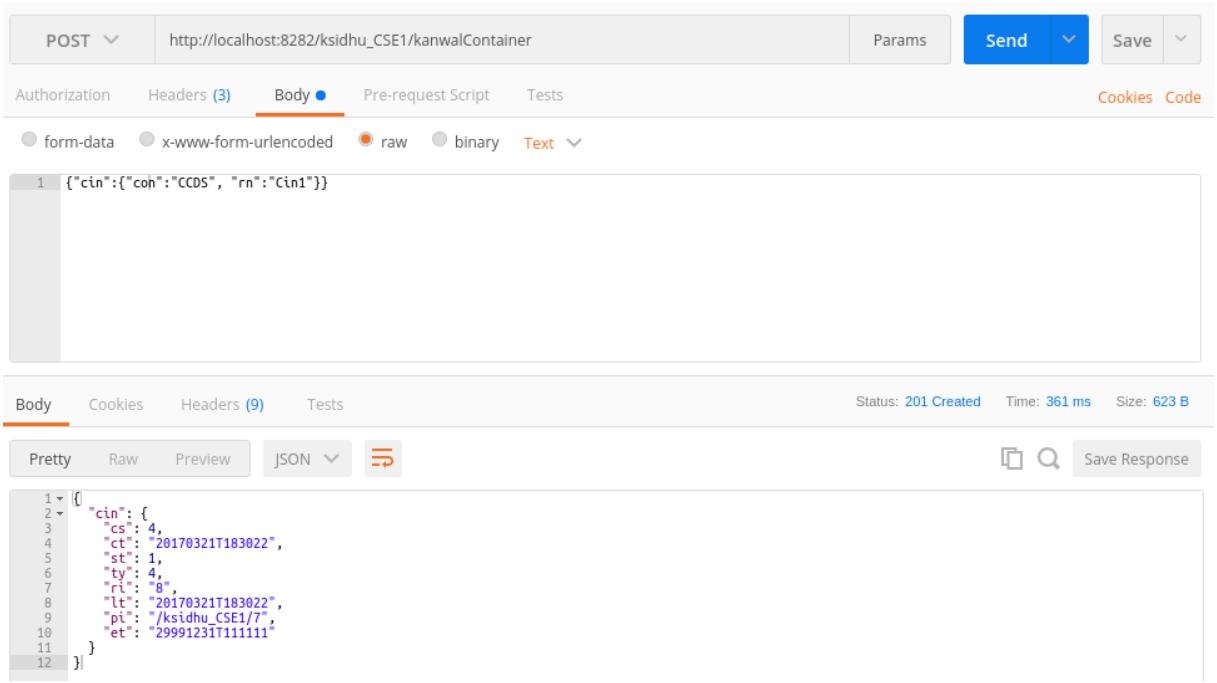
TEST 11:

Create a Content Instance: Cin1 in kanwalContainer.

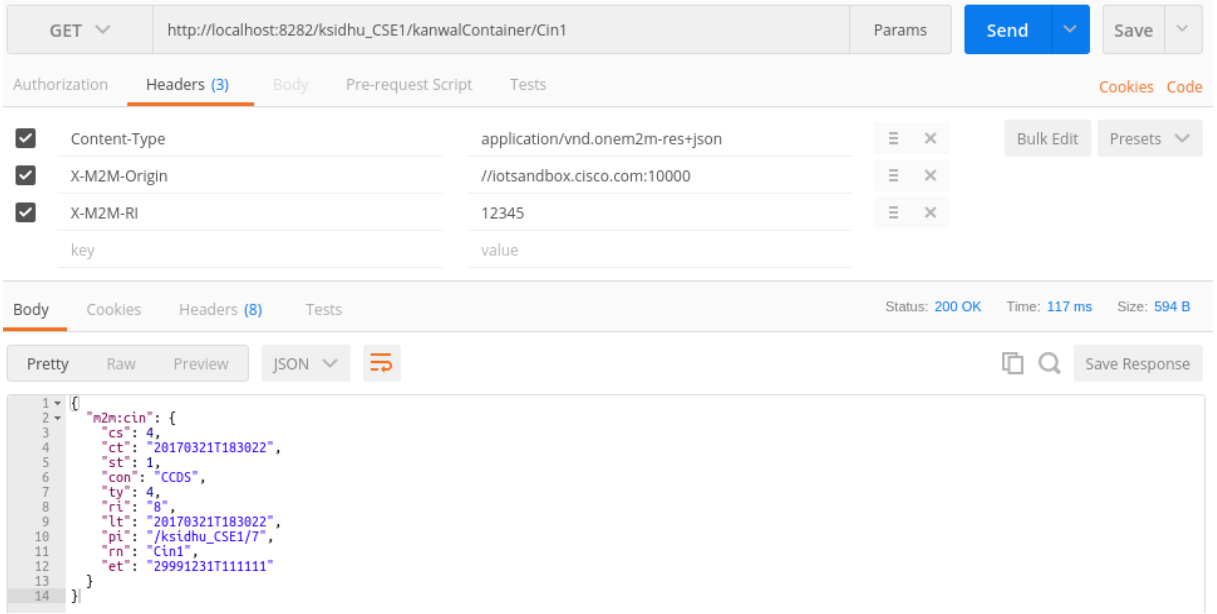
POST http://localhost:8282/ksidhu_CSE1/kanwalContainer

RESPONSE:

ContentInstance:Cin1 has been successfully created in kanwalContainer and it could be retrieved with ResourceName(rn) as Cin1 along with other attributes.



Retrieve contentinstance



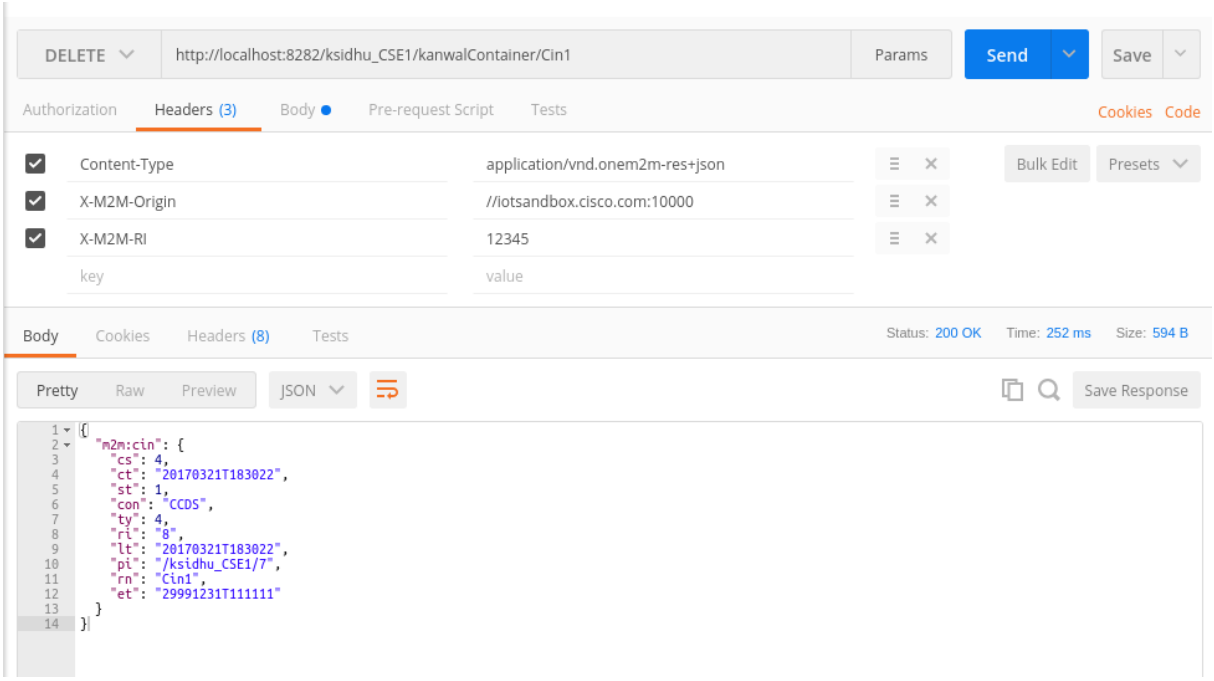
TEST 12:

Delete a Content Instance: Cin1 in kanwalContainer.

DELETE http://localhost:8282/ksidhu_CSE1/kanwalContainer/Cin1

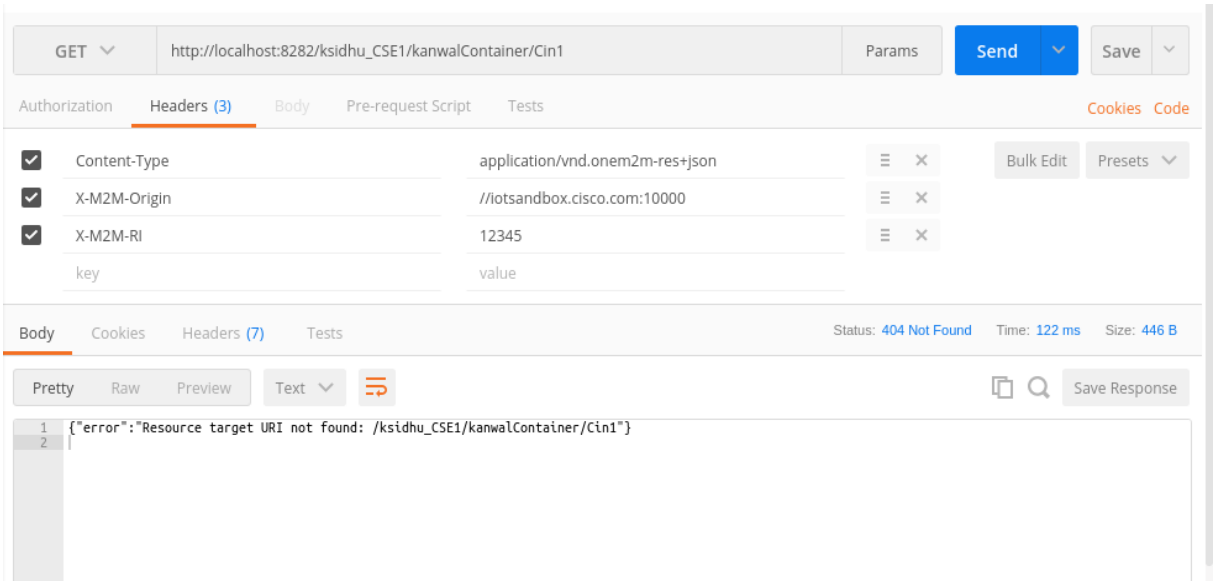
RESPONSE:

ContentInstance with (rn): Cin1 has been successfully deleted in kanwalContainer and we can see that target URI not found in the retrieve request.



Retrieve after deleting

GET http://localhost:8282/ksidhu_CSE1/kanwalContainer/Cin1



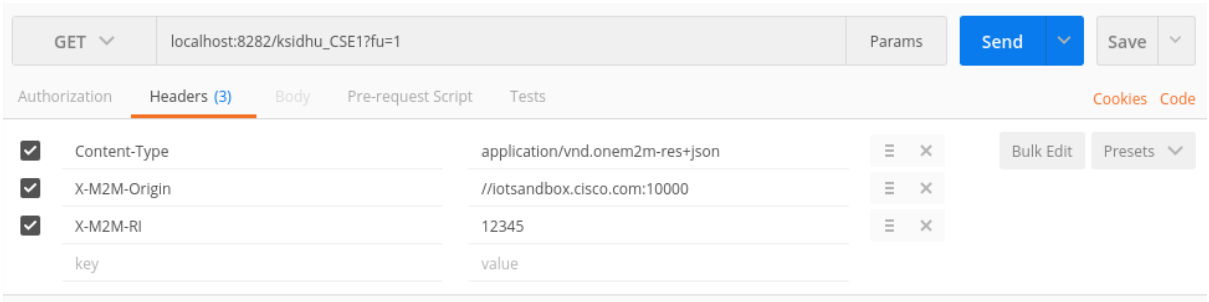
TEST 13:

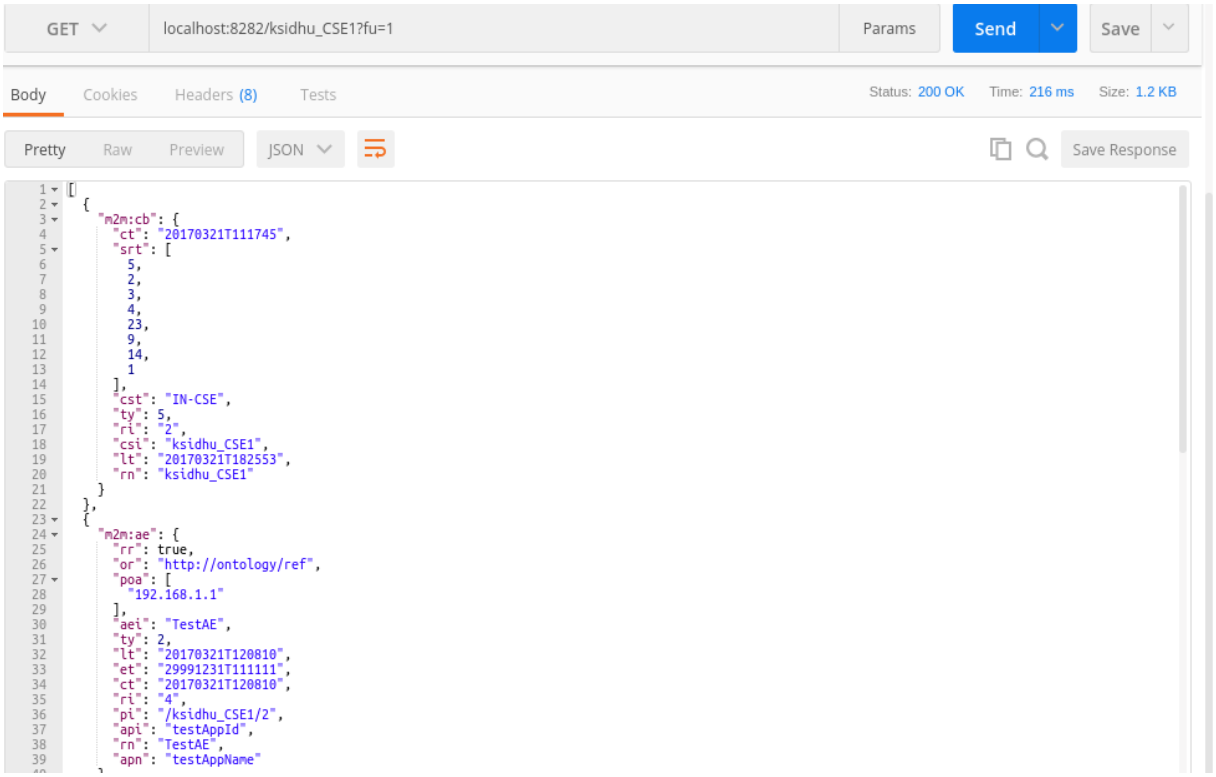
Discover all the elements of the resource tree

GET http://localhost:8282/ksidhu_CSE1?fu=1

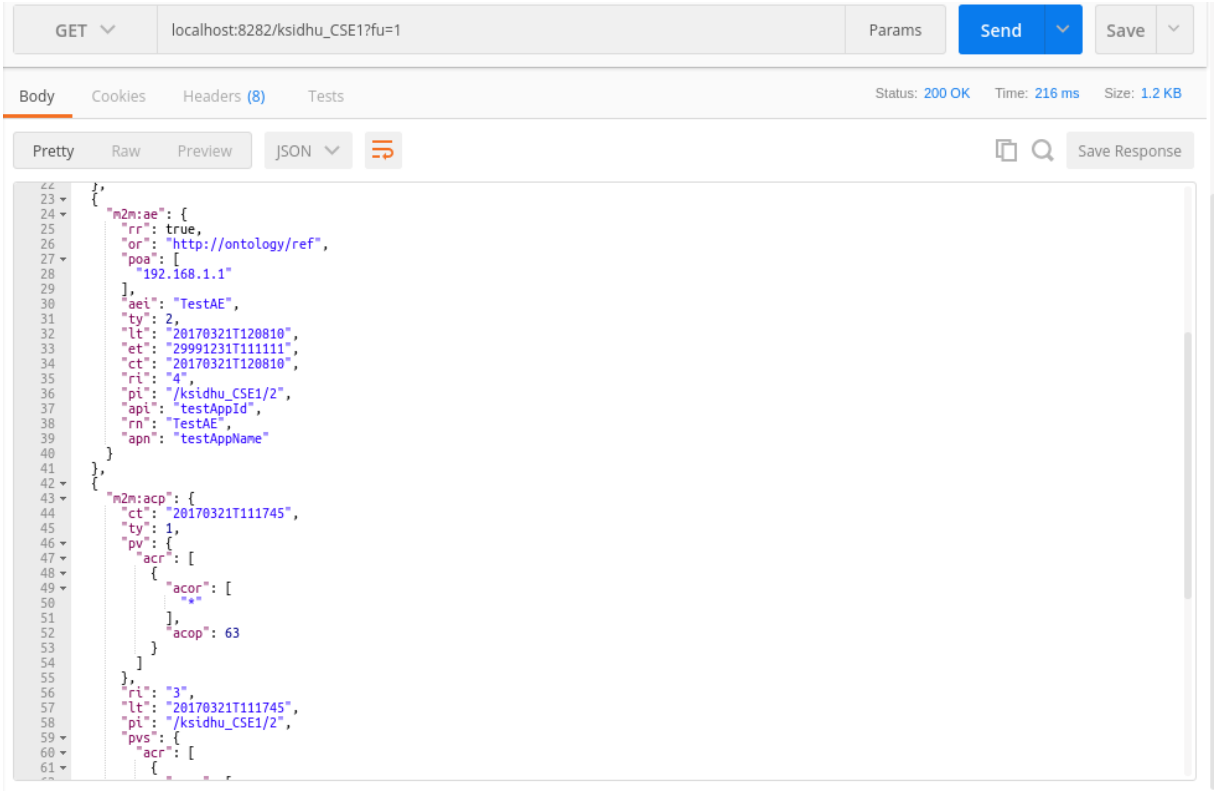
RESPONSE:

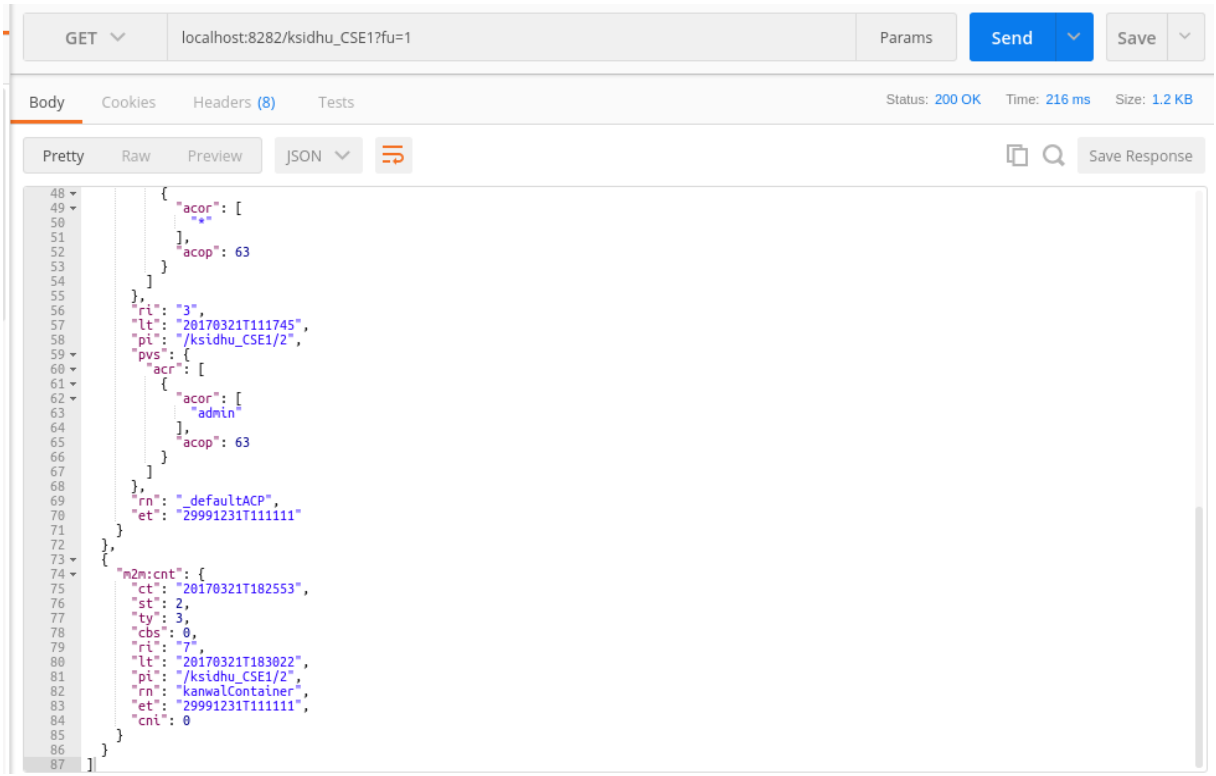
This is a great way to keep track of the resource tree or to discover all the children. In the response window we could see full detail of resources CSE, AE, ACP, Container along with their respective attributes. Long response containing all the information is shown in various screenshots below. We can still retrieve the individual responses for each resource but it's a good way to navigate into the already created tree.






```
[
  {
    "m2m:cb": {
      "ct": "20170321T111745",
      "srt": [
        5,
        2,
        3,
        4,
        23,
        9,
        14,
        1
      ],
      "cst": "IN-CSE",
      "ty": 5,
      "ri": "2",
      "csi": "ksidhu_CSE1",
      "lt": "20170321T182553",
      "rn": "ksidhu_CSE1"
    }
  },
  {
    "m2m:ae": {
      "rr": true,
      "or": "http://ontology/ref",
      "poa": [
        "192.168.1.1"
      ],
      "aei": "TestAE",
      "ty": 2,
      "lt": "20170321T120810",
      "et": "29991231T111111",
      "ct": "20170321T120810",
      "ri": "4",
      "pi": "/ksidhu_CSE1/2",
      "api": "testAppId",
      "rn": "TestAE",
      "apn": "testAppName"
```


GET
localhost:8282/ksidhu_CSE1?fu=1
Params
Send
Save
Body
Cookies
Headers (8)
Tests
Status: 200 OK
Time: 216 ms
Size: 1.2 KB
Pretty
Raw
Preview
JSON
Save Response
},
{
"m2m:ae": {
"rr": true,
"or": "http://ontology/ref",
"poa": [
"192.168.1.1"
],
"aei": "TestAE",
"ty": 2,
"lt": "20170321T120810",
"et": "29991231T111111",
"ct": "20170321T120810",
"ri": "4",
"pi": "/ksidhu_CSE1/2",
"api": "testAppId",
"rn": "TestAE",
"apn": "testAppName"
}
},
{
"m2m:acp": {
"ct": "20170321T111745",
"ty": 1,
"pv": {
"acr": [
{
"acor": [
"*"
],
"acop": 63
}
]
},
"ri": "3",
"lt": "20170321T111745",
"pi": "/ksidhu_CSE1/2",
"pvs": {
"acr": [
{


GET
localhost:8282/ksidhu_CSE1?fu=1
Params
Send
Save
Body
Cookies
Headers (8)
Tests
Status: 200 OK
Time: 216 ms
Size: 1.2 KB
Pretty
Raw
Preview
JSON
Save Response
{
"acor": [
"*"
],
"acop": 63
}
]
},
"ri": "3",
"lt": "20170321T111745",
"pi": "/ksidhu_CSE1/2",
"pvs": {
"acr": [
{
"acor": [
"admin"
],
"acop": 63
}
]
},
"rn": "_defaultACP",
"et": "29991231T111111"
}
},
{
"m2m:cnt": {
"ct": "20170321T182553",
"st": 2,
"ty": 3,
"cbs": 0,
"ri": "7",
"lt": "20170321T183022",
"pi": "/ksidhu_CSE1/2",
"rn": "kanwalContainer",
"et": "29991231T111111",
"cni": 0
}
}
]

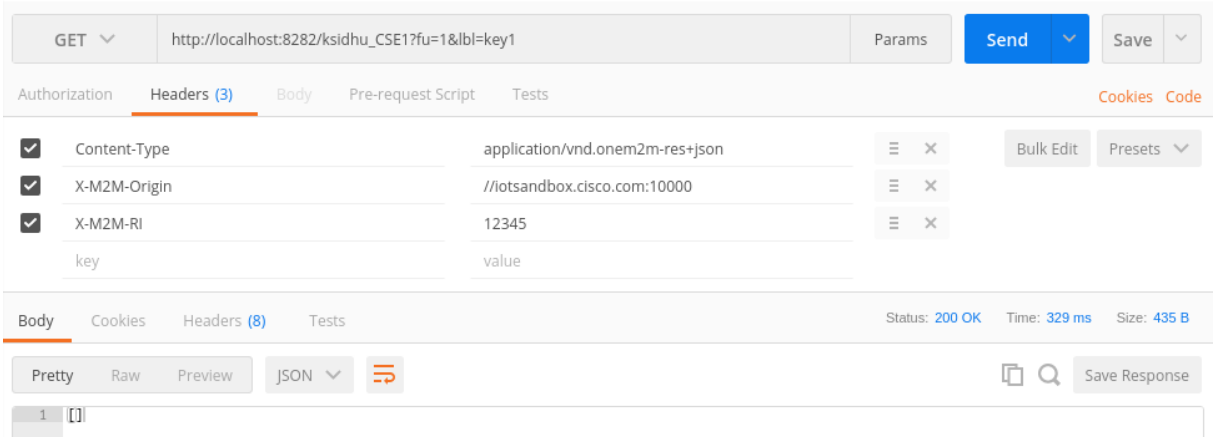
TEST 14:

Discover all the resources with Label as key1

GET http://localhost:8282/ksidhu_CSE1?fu=1&lbl=key1

RESPONSE:

This is a good way to track all the resources with any common label. As we don't have this label in any of our resources therefore response window is empty.



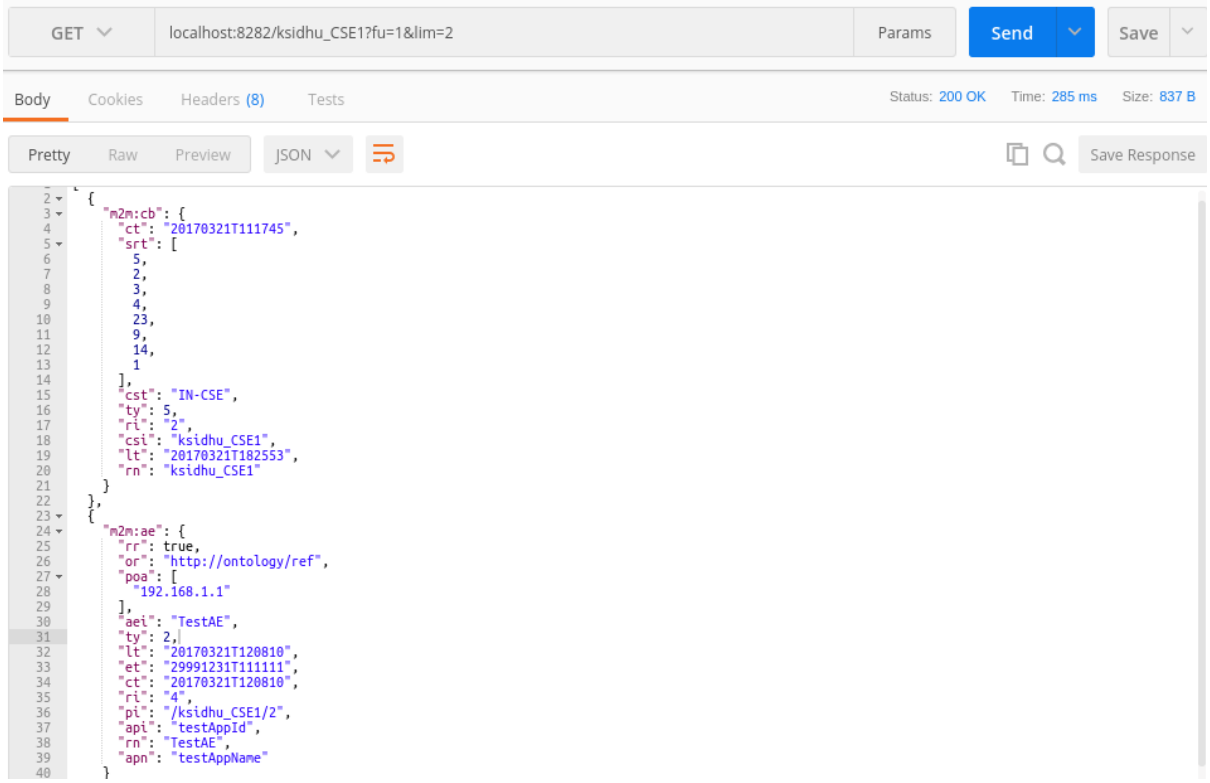
TEST 15:

Discover resources but limit the response to 2.

GET http://localhost:8282/ksidhu_CSE1?fu=1&lim=2

RESPONSE:

If we don't want to discover the full resource tree and only want to check the top resources we can limit our output to any limit. In my case, I limited the response to 2, which gave me top two resources CSE: ksidhu_CSE1 and AE: testAE which we created earlier.



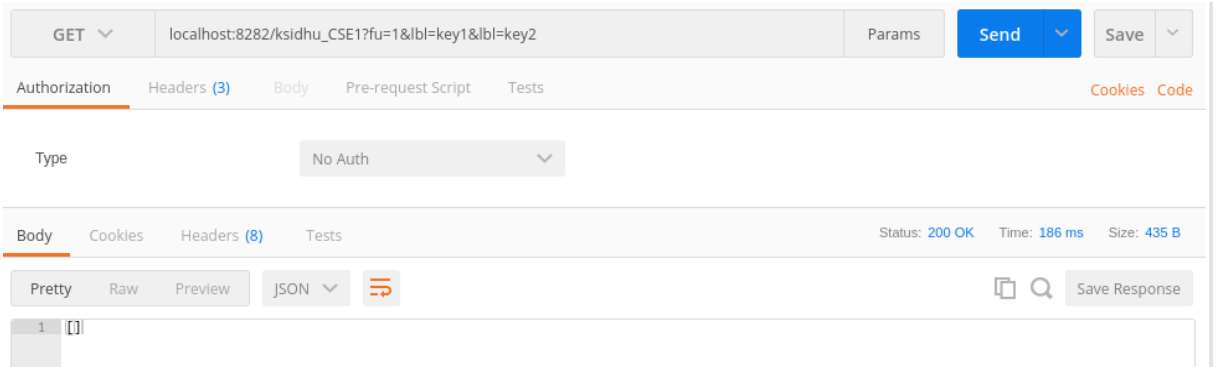
TEST 16:

Discover all the resources with label as key1 and key2

GET http://localhost:8282/ksidhu_CSE1?fu=1&lbl=key1&lbl=key2

RESPONSE:

We can also discover our resources with multiple criteria as shown in this Test.



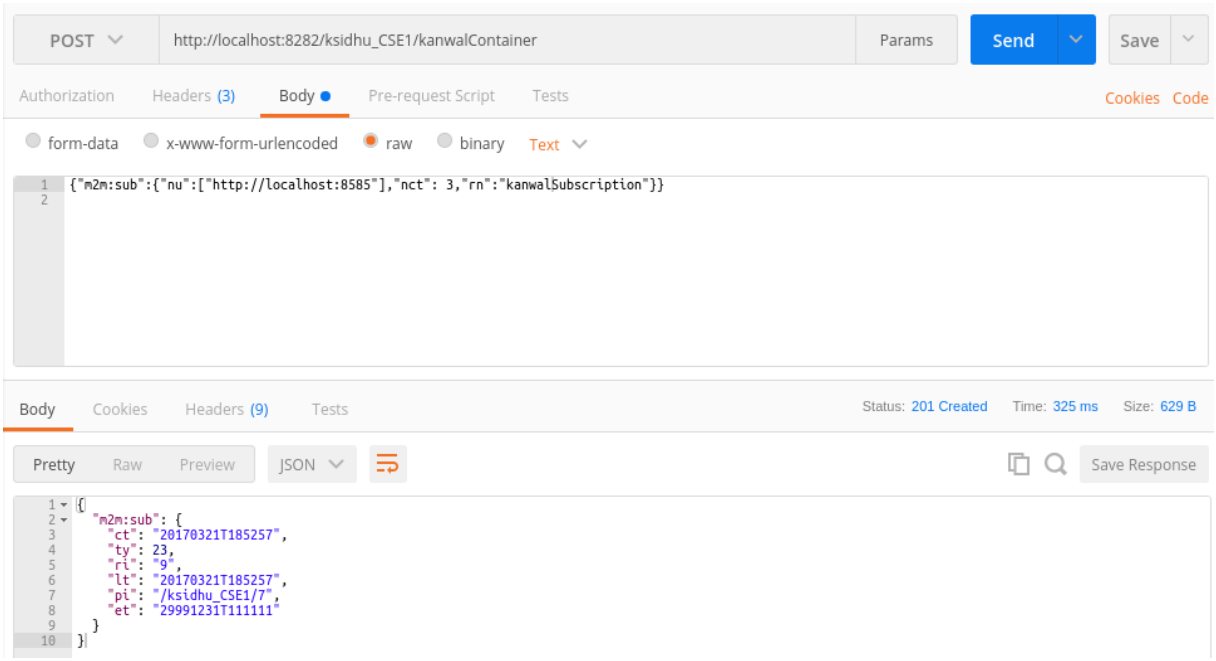
TEST 17:

Create a new subscription with ResourceName: kanwalSubscription.

POST http://localhost:8282/ksidhu_CSE1/kanwalContainer

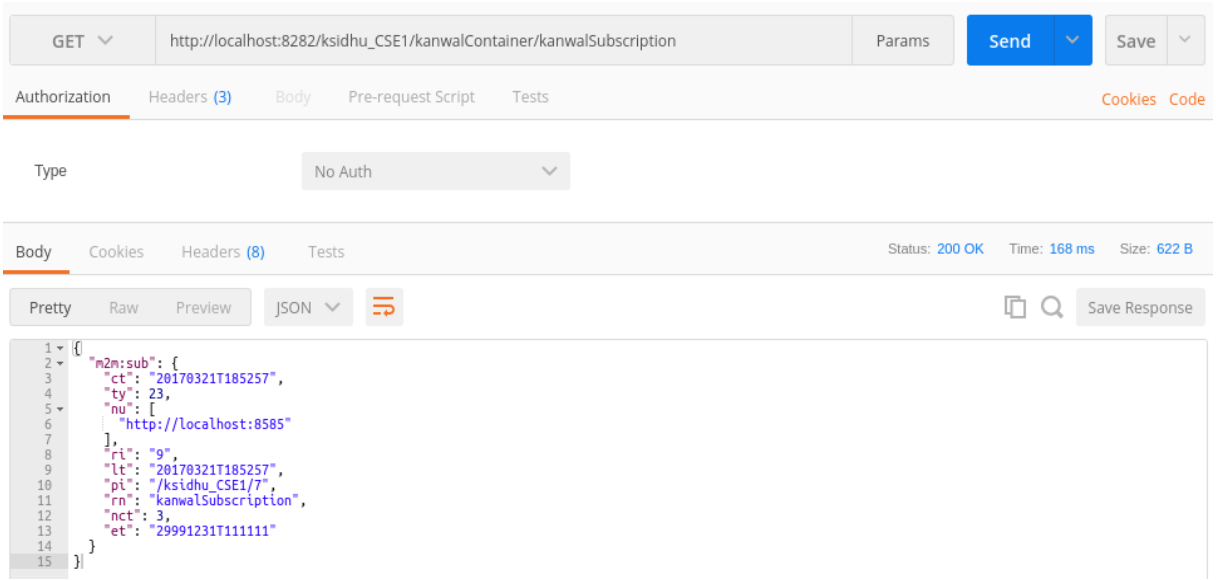
RESPONSE:

In this Test, I have successfully created a new subscription as kanwalSunscription under kanwalContainer.



Retrieve kanwalSubscription

GET http://localhost:8282/ksidhu_CSE1/kanwalContainer/kanwalSubscription



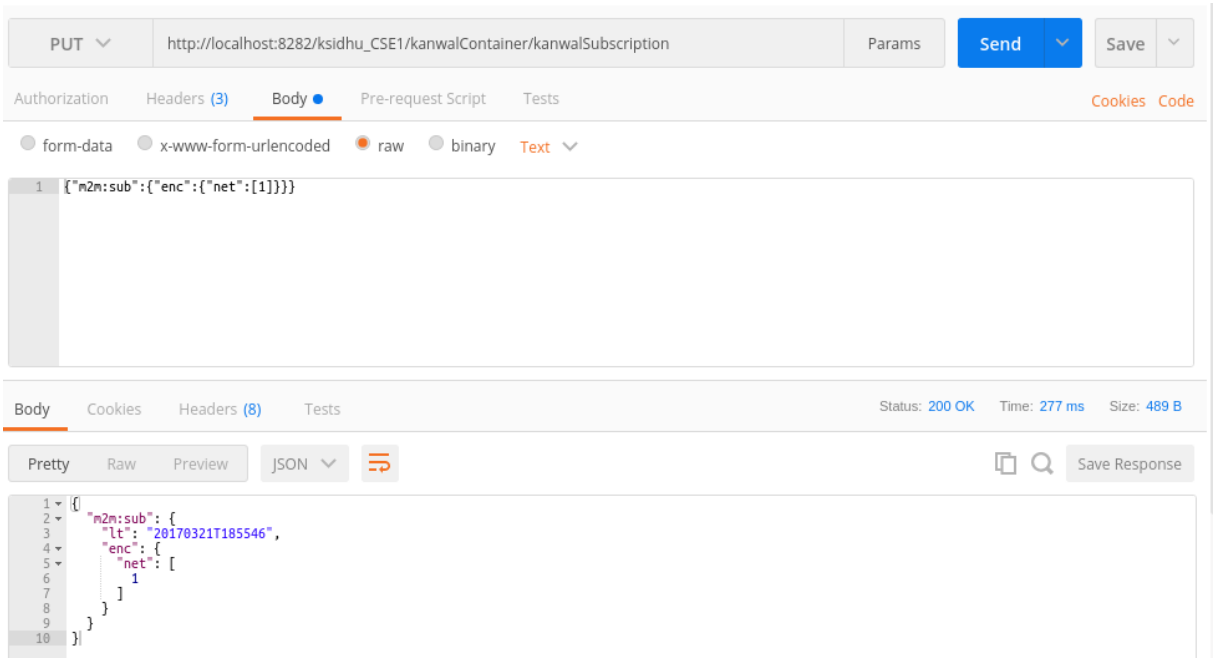
TEST 18:

Update subscription with ResourceName: kanwalSubscription.

PUT http://localhost:8282/ksidhu_CSE1/kanwalContainer/kanwalSubscription

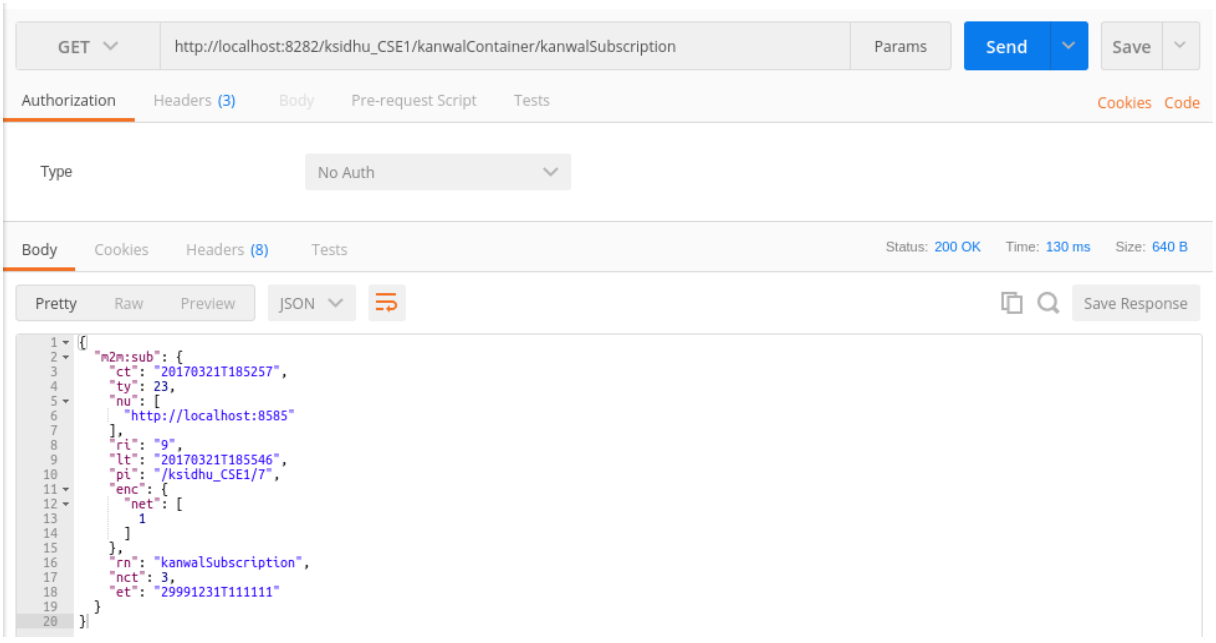
RESPONSE:

In this Test, I have successfully updated a new subscription: kanwalSunscription under kanwalContainer with net attribute as 1. This updated attributes could be seen in the retrieved subscription.



Retrieve kanwalSubscription after update

GET http://localhost:8282/ksidhu_CSE1/kanwalContainer/kanwalSubscription



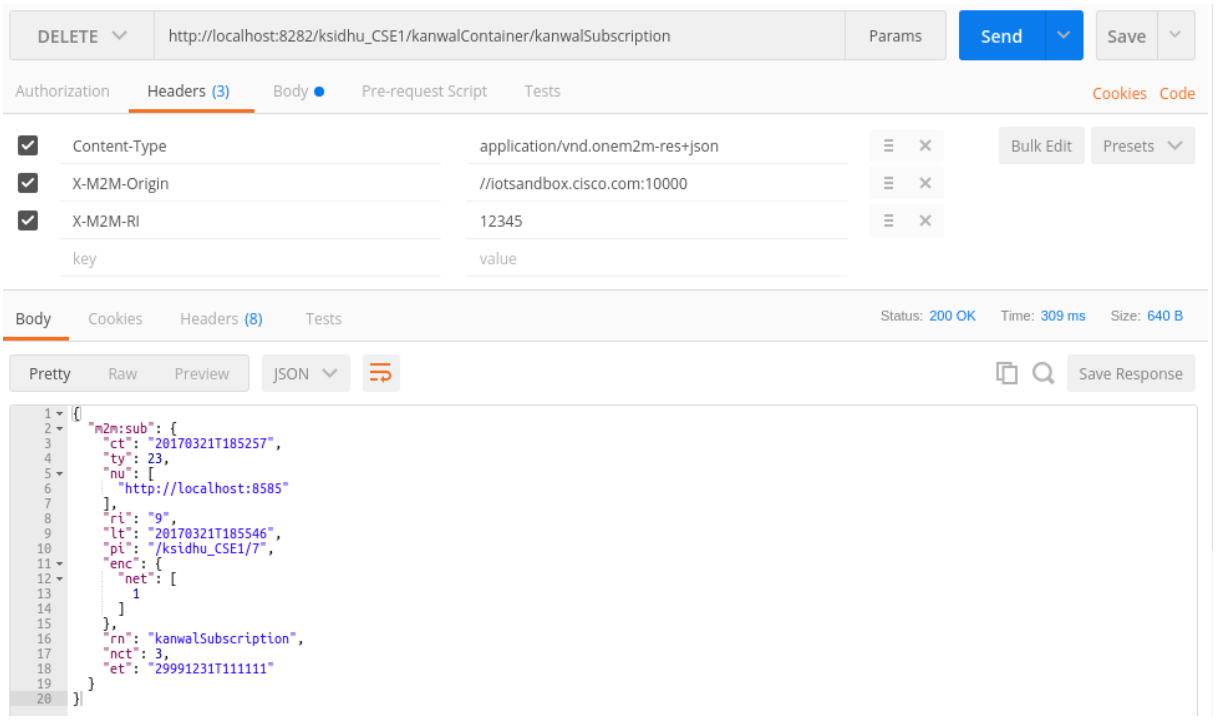
TEST 19:

Delete subscription with ResourceName: kanwalSubscription.

DELETE http://localhost:8282/ksidhu_CSE1/kanwalContainer/kanwalSubscription

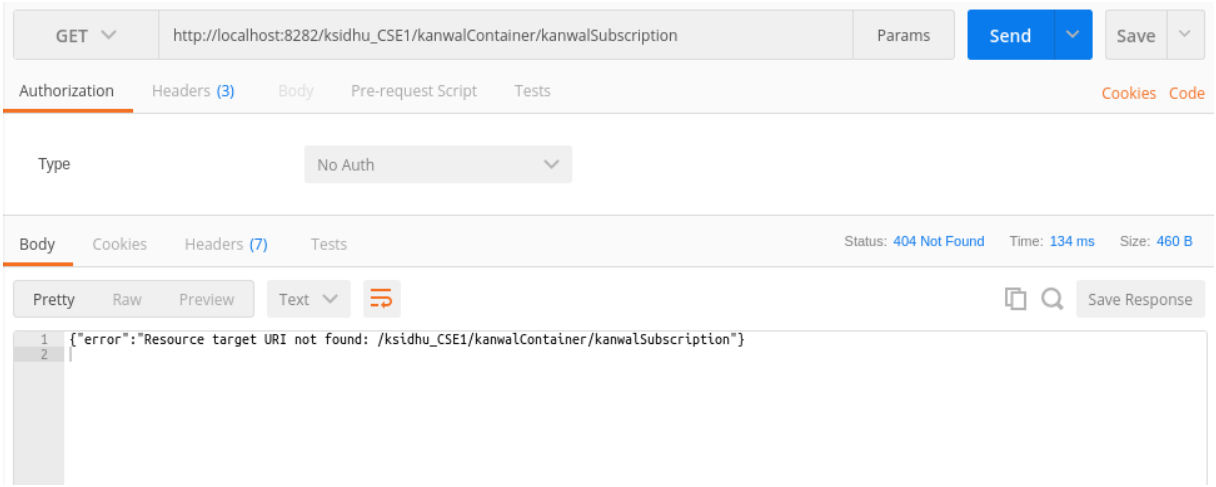
RESPONSE:

In this Test, I have successfully deleted subscription: kanwalSunscription under kanwalContainer and Retrieve after delete response shows that Target URI not found.



Retrieve kanwalSubscription after delete

GET http://localhost:8282/ksidhu_CSE1/kanwalContainer/kanwalSubscription



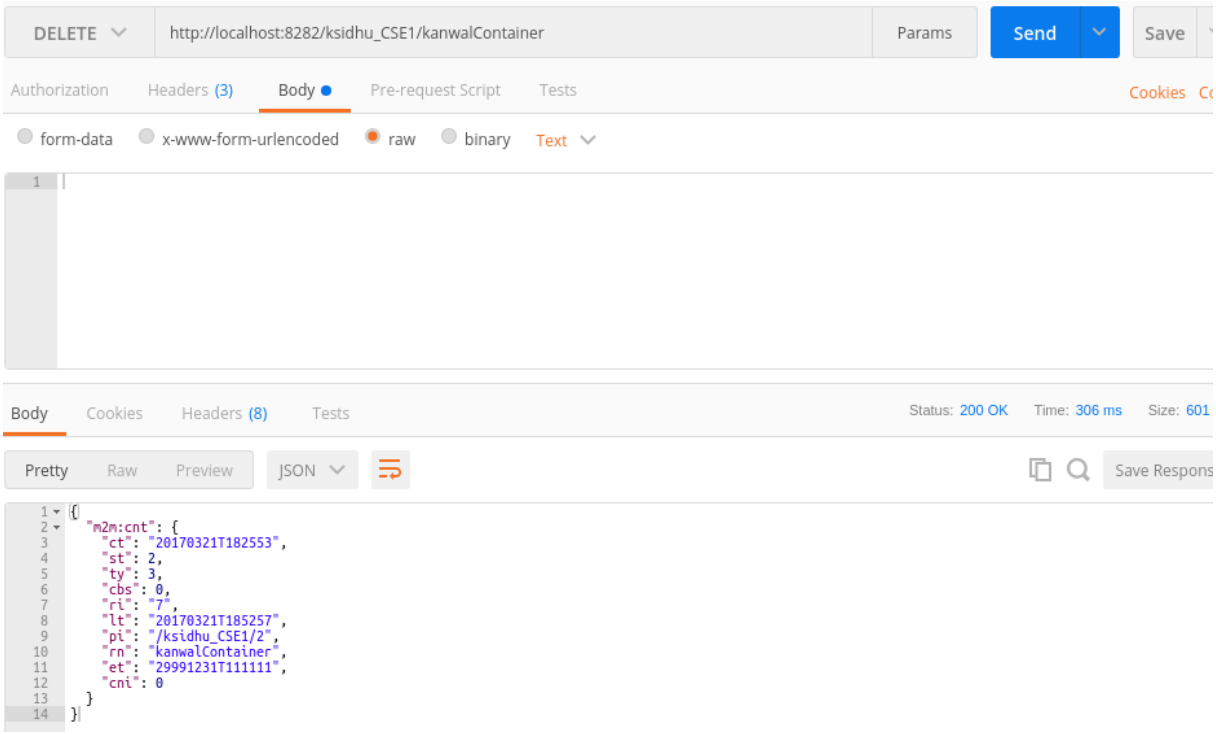
TEST 20:

Delete Container with ResourceName: kanwalContainer.

DELETE http://localhost:8282/ksidhu_CSE1/kanwalContainer

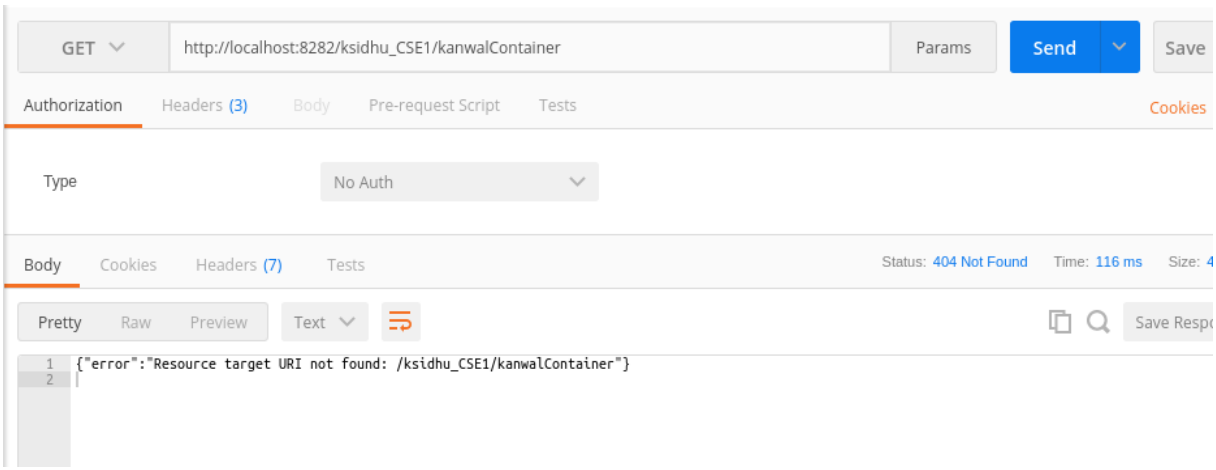
RESPONSE:

In this Test, I have successfully deleted Container: kanwalContainer and Retrieve after delete response shows that Target URI not found.



Retrieve kanwalContainer after delete

GET http://localhost:8282/ksidhu_CSE1/kanwalContainer

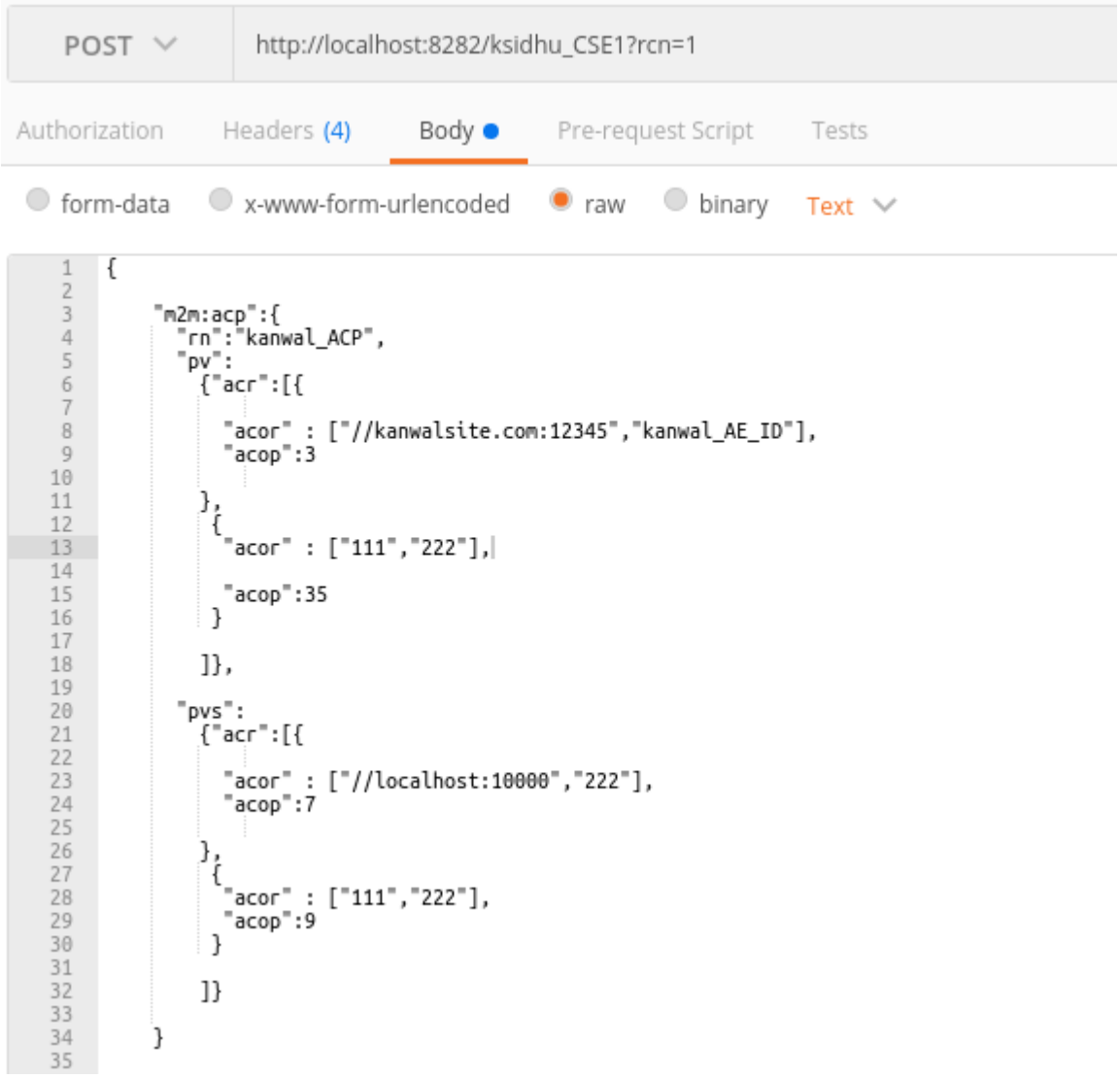


TEST 21:

Create Access Control Policy.

POST http://localhost:8282/ksidhu_CSE1?rcn=1

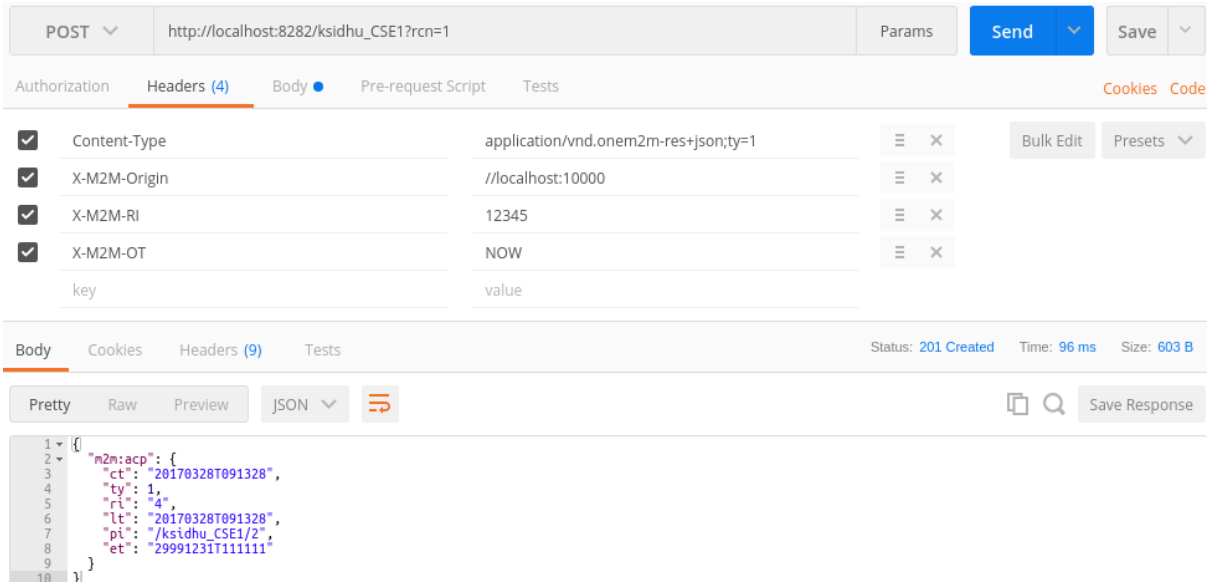
Body:



Access Control Originator (acor) kanwal_AE_ID can perform a set of operations under acop:3. Access Control Originator (acor) 222 can perform a set of operations under acop:7. We can update these values anytime based upon the requirement.

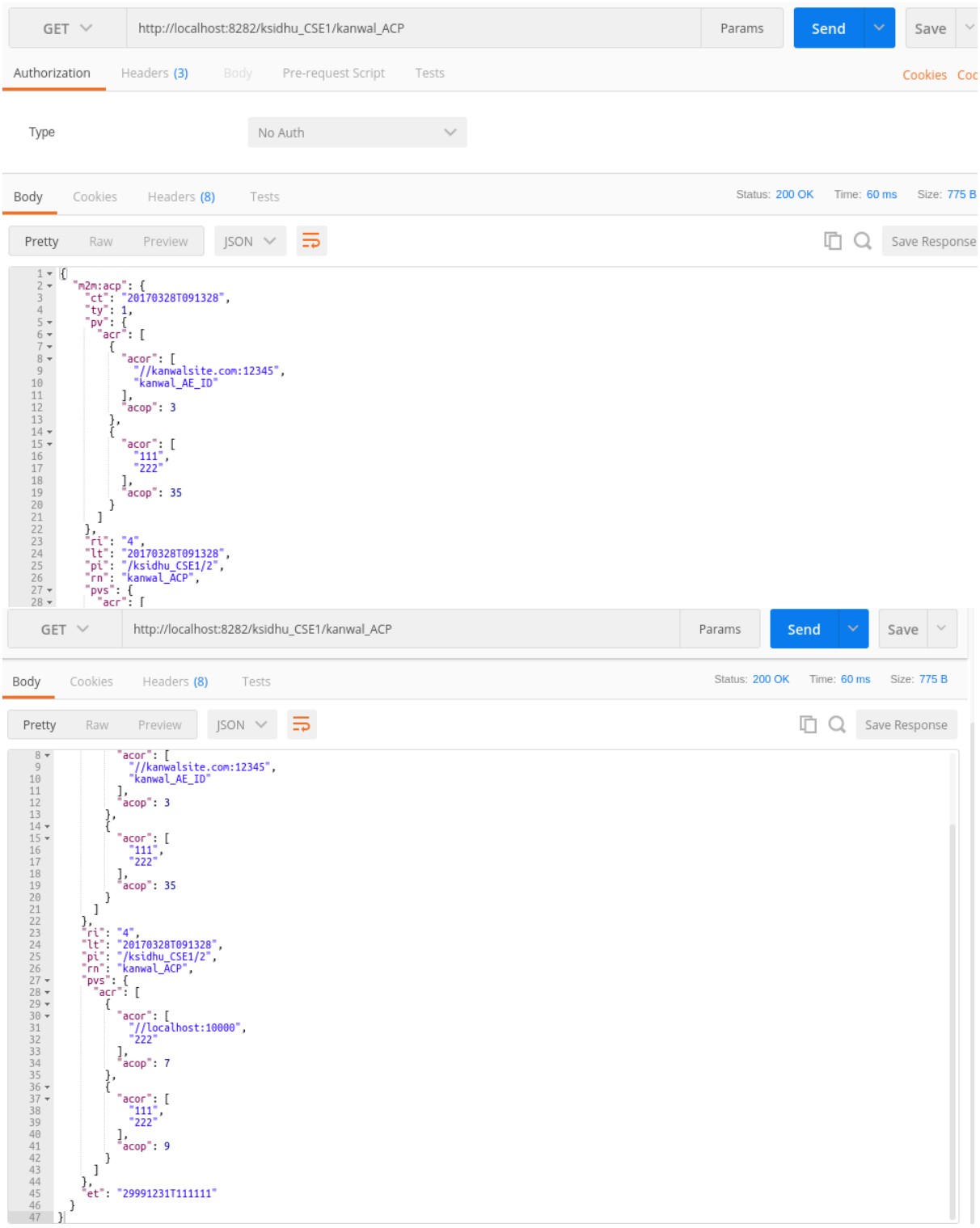
RESPONSE:

In this Test, I have successfully created AccessControlPolicy kanwal_ACP containing attributes of the Body of the POST request.



Retrieve kanwal_ACP after create

GET http://localhost:8282/ksidhu_CSE1/kanwal_ACP

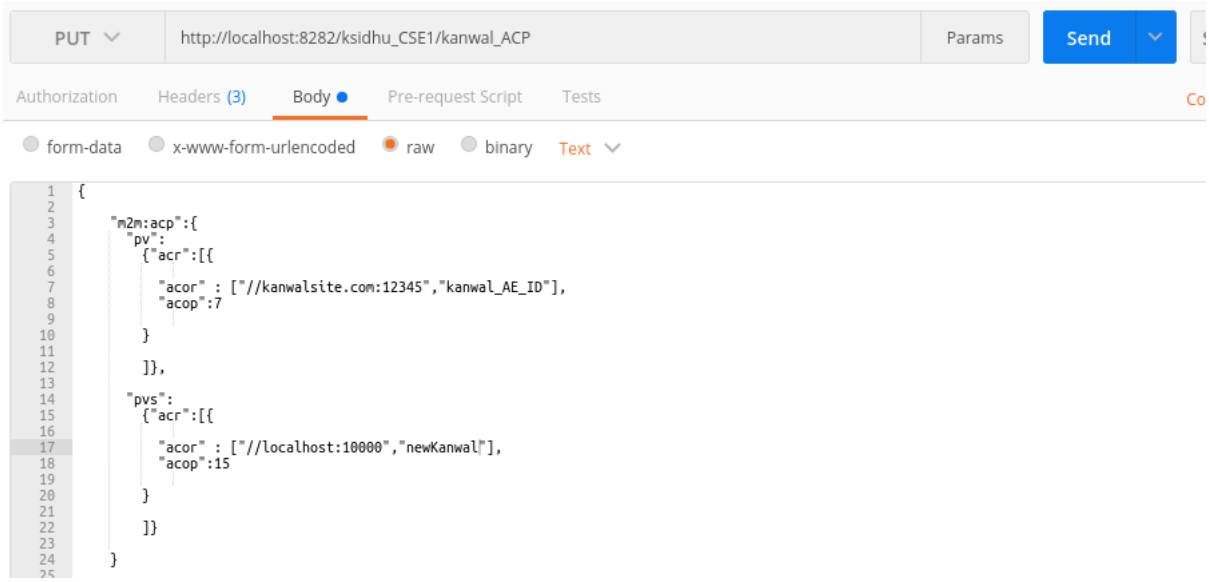


TEST 22:

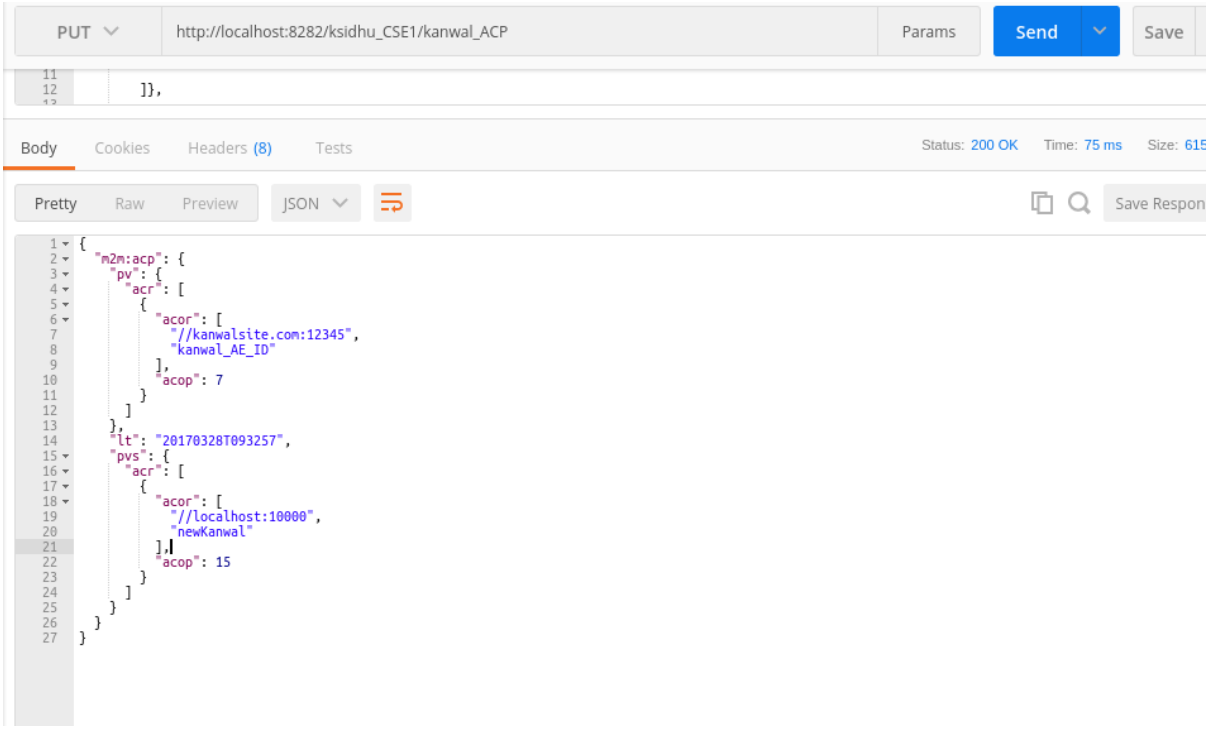
Update recently created Access Control Policy.

POST http://localhost:8282/ksidhu_CSE1?rcn=1

Body:



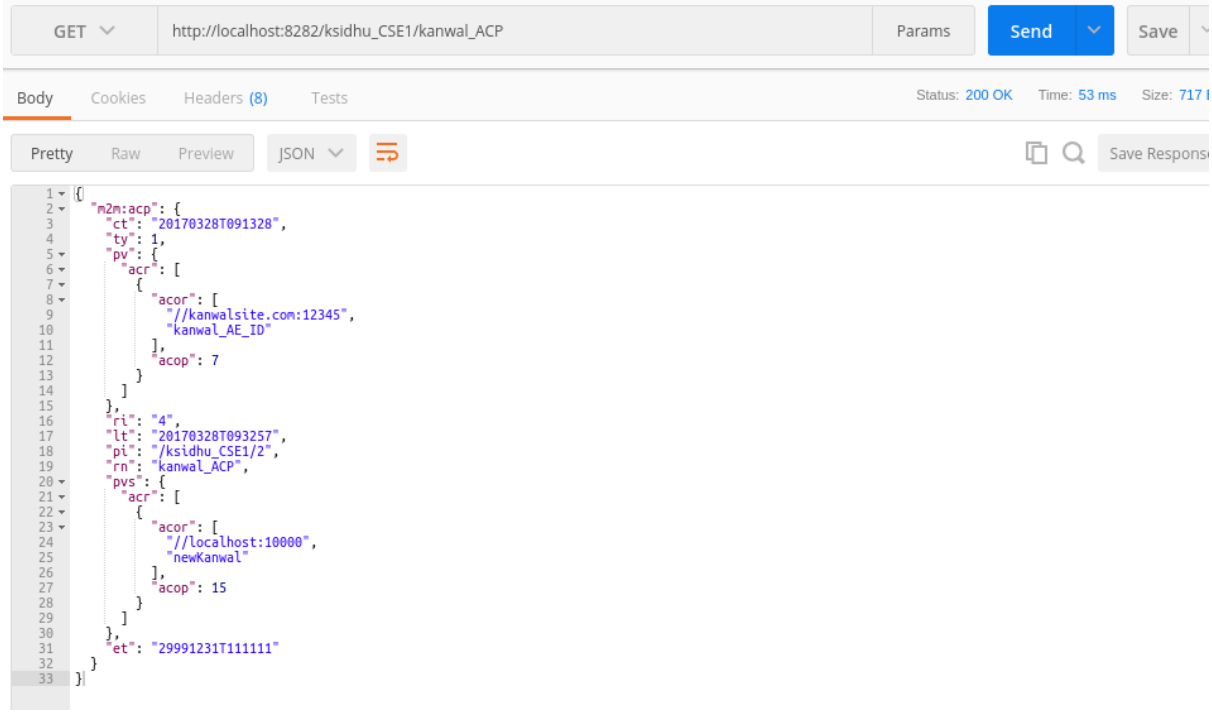
RESPONSE:



Retrieve kanwal_ACP after update

GET http://localhost:8282/ksidhu_CSE1/kanwal_ACP

We can see the updated attributes acor : kanwal_AE_ID with new acop value as 7. Similarily newKanwal is also updated with new acop:15



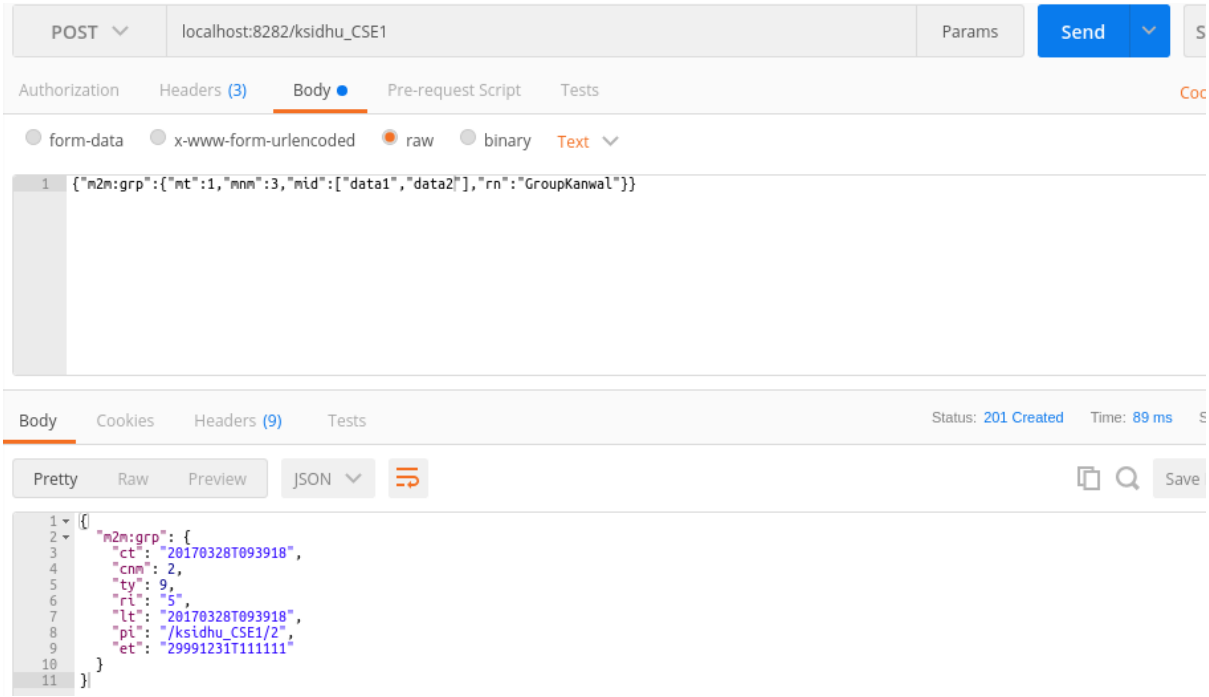
TEST 23:

Create a new Group ResourceName: GroupKanwal.

POST http://localhost:8282/ksidhu_CSE1

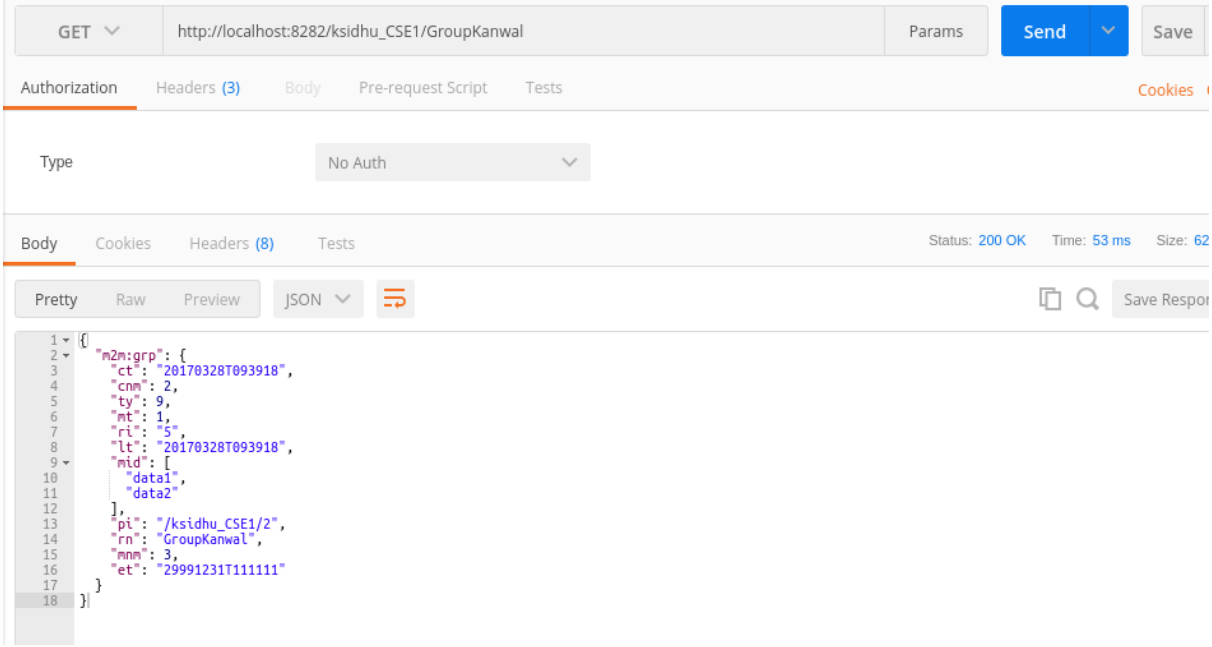
RESPONSE:

In this Test, I have successfully created a new Group: GroupKanwal under ksidhu_CSE1 with miscellaneous data (mid) as data1 and data2 just arbitrary values.



Retrieve GroupKanwal

GET http://localhost:8282/ksidhu_CSE1/GroupKanwal



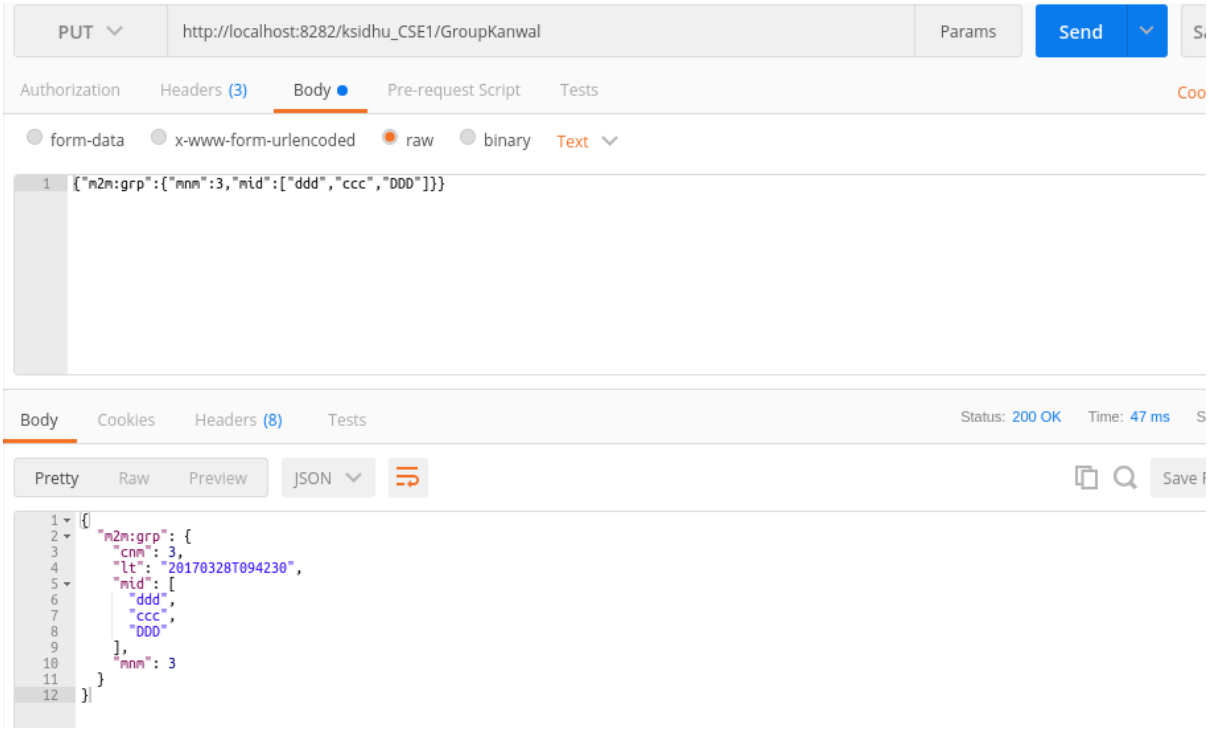
TEST 24:

Update Group with ResourceName: GroupKanwal.

POST http://localhost:8282/ksidhu_CSE1/GroupKanwal

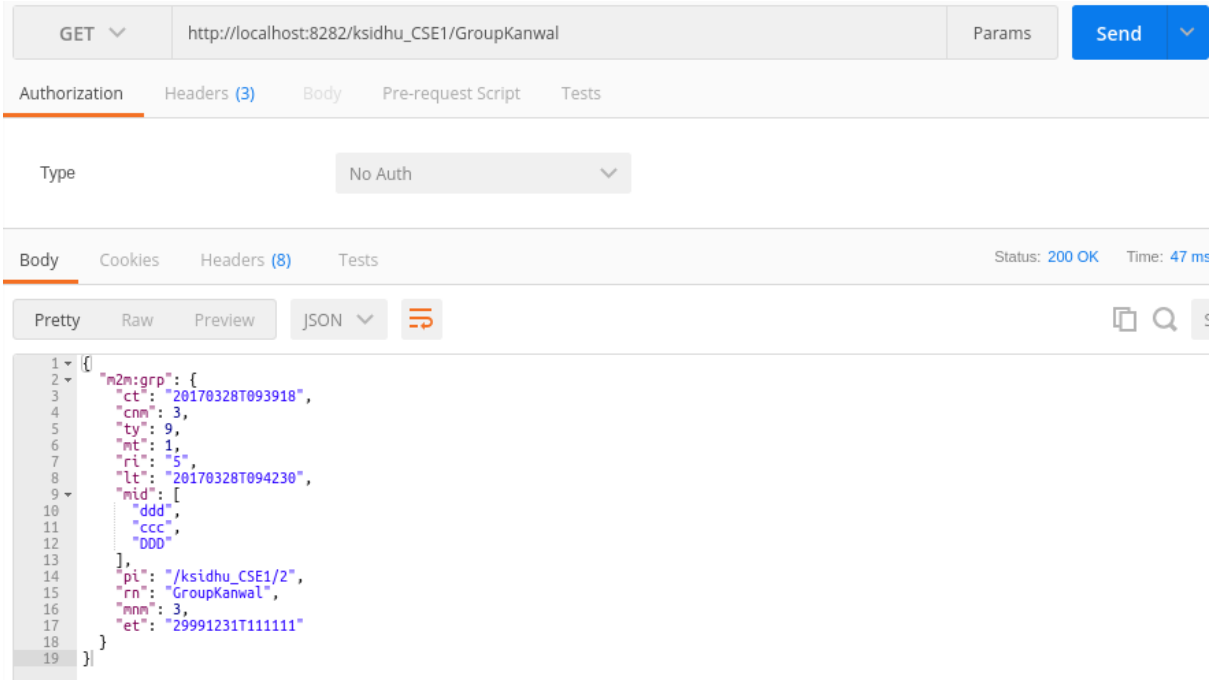
RESPONSE:

In this Test, I have successfully updated GroupKanwal under ksidhu_CSE1 with miscellaneous data (mid) as ddd, ccc and DDD just arbitrary values. We can observe the new values in the retrieve response window.



Retrieve GroupKanwal

GET http://localhost:8282/ksidhu_CSE1/GroupKanwal



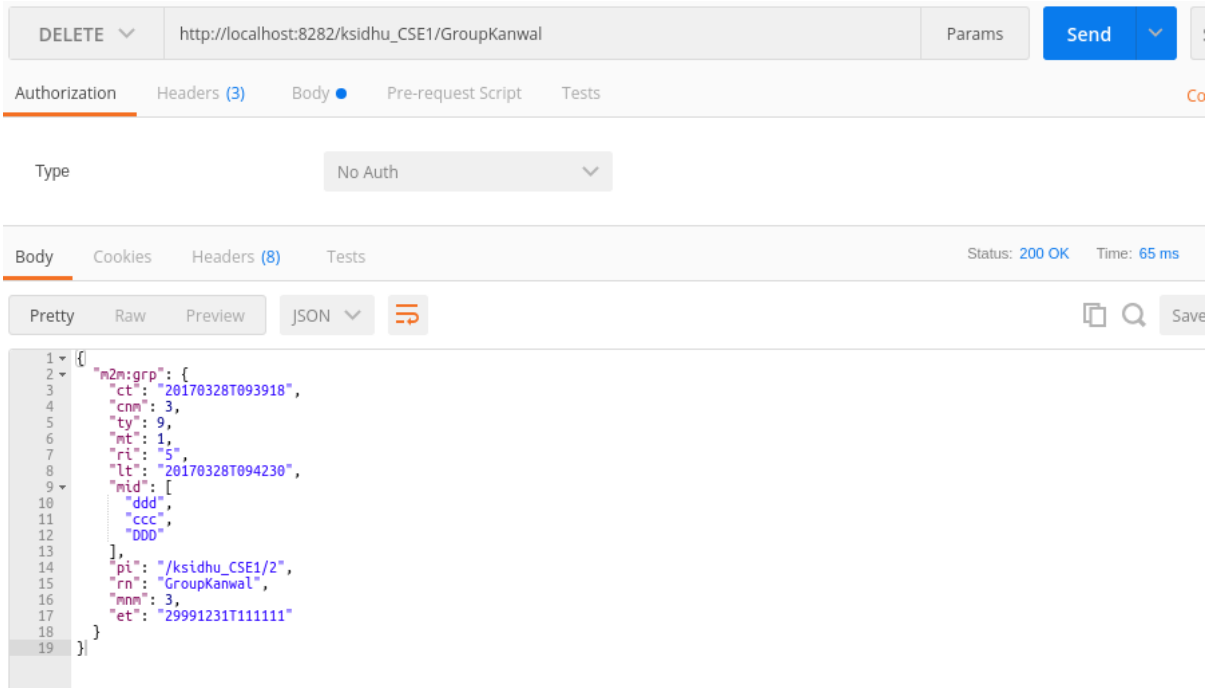
TEST 25:

Delete Group with ResourceName: GroupKanwal.

DELETE http://localhost:8282/ksidhu_CSE1/GroupKanwal

RESPONSE:

In this Test, I have successfully deleted GroupKanwal under ksidhu_CSE1 and Retrieve after delete response shows that Target URI not found.



Retrieve GroupKanwal

GET http://localhost:8282/ksidhu_CSE1/GroupKanwal

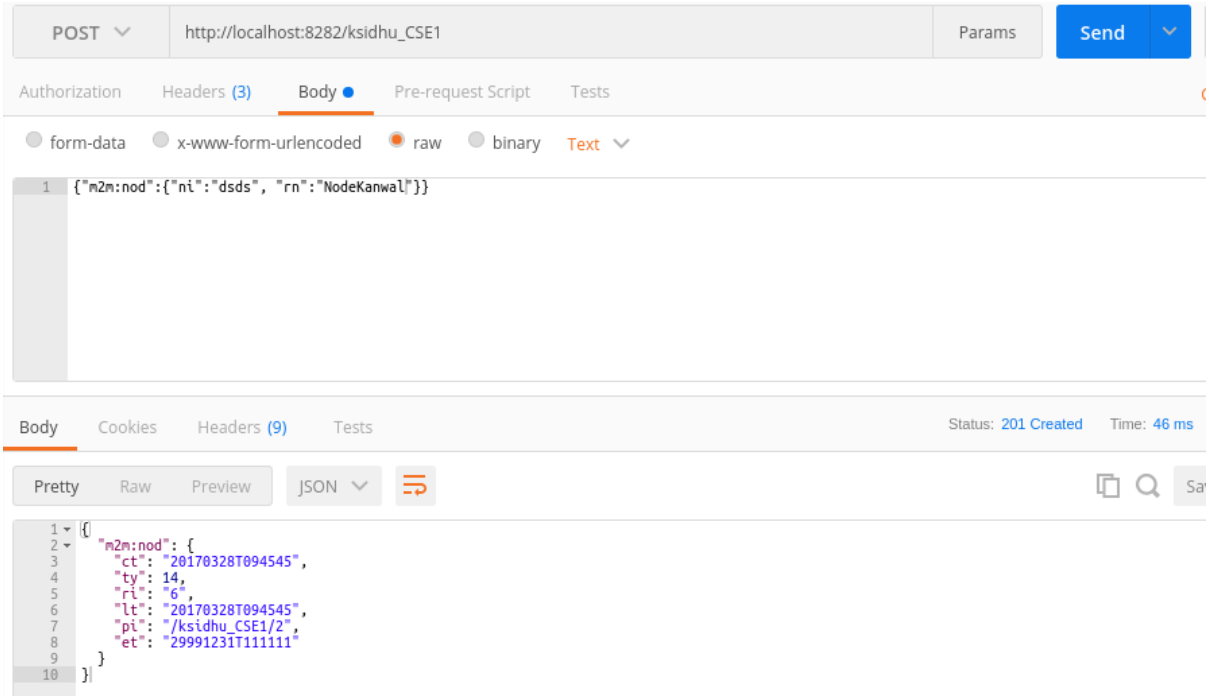
TEST 26:

Create a new Resource Node with ResourceName: NodeKanwal.

POST http://localhost:8282/ksidhu_CSE1

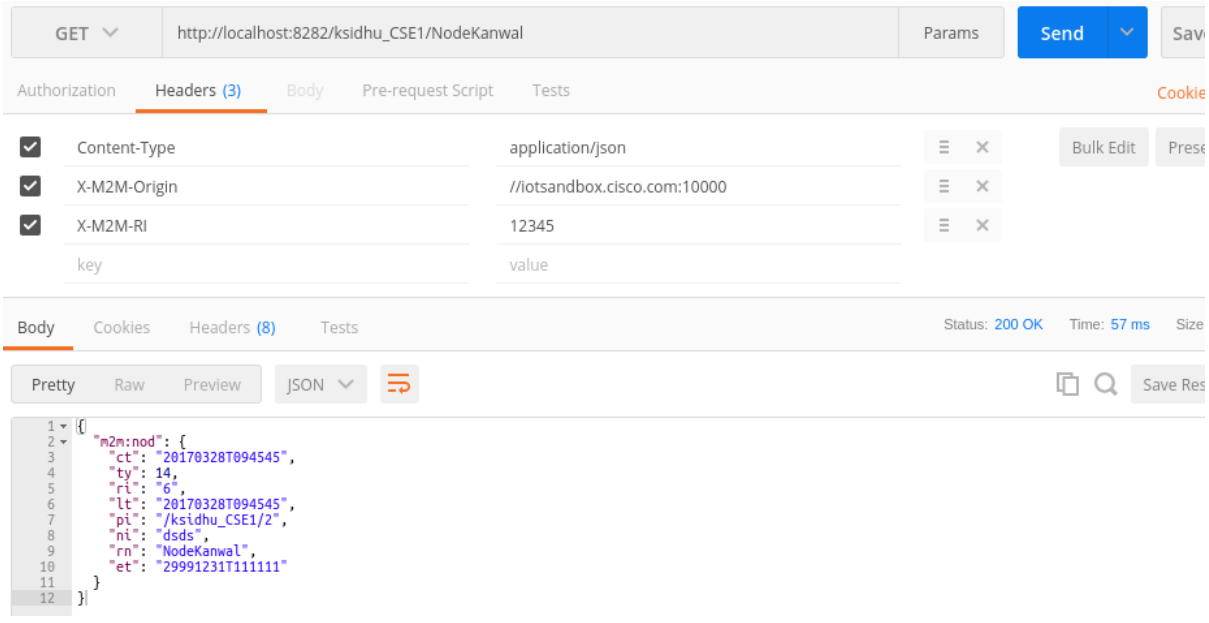
RESPONSE:

In this Test, I have successfully created a new resource Node: NodeKanwal under ksidhu_CSE1 and this node can be retrieved in by retrieve request as shown.



Retrieve NodeKanwal

GET http://localhost:8282/ksidhu_CSE1/NodeKanwal



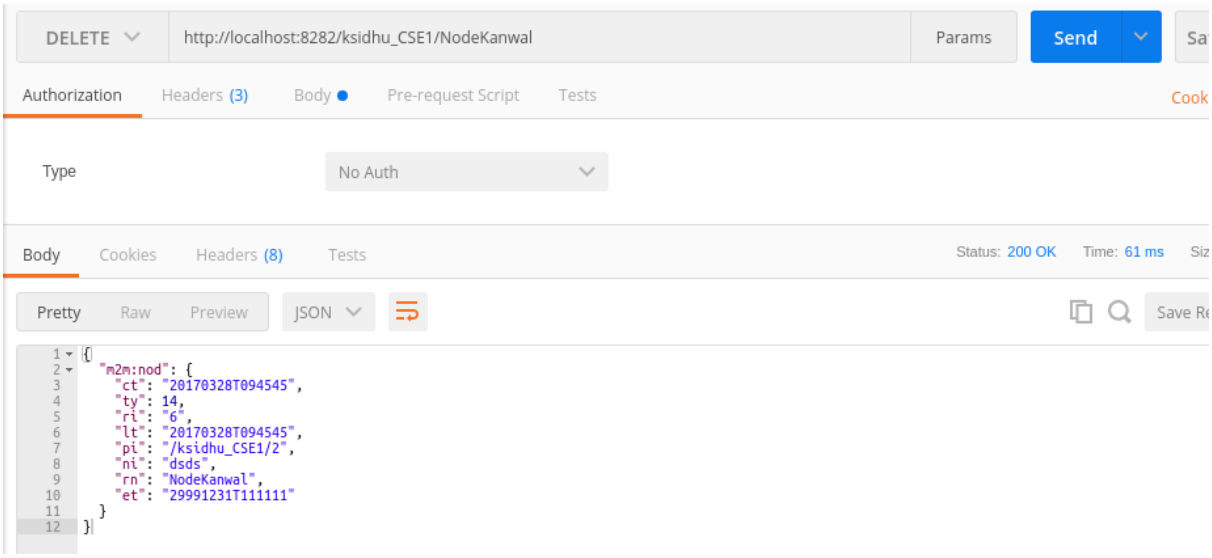
TEST 27:

Delete Node with ResourceName: NodeKanwal.

DELETE http://localhost:8282/ksidhu_CSE1/NodeKanwal

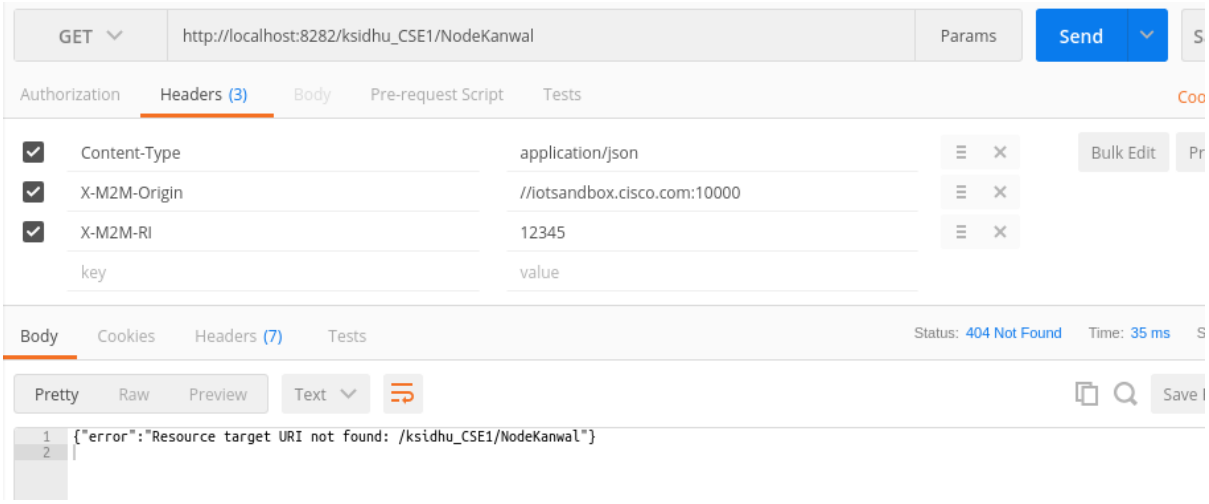
RESPONSE:

In this Test, I have successfully deleted NodeKanwal under ksidhu_CSE1 and Retrieve after delete response shows that Target URI not found.



Retrieve NodeKanwal

GET http://localhost:8282/ksidhu_CSE1/NodeKanwal



5 Summary

To implement Remote Procedure Calls for building new applications for the IoTDM systems, this Testing is the key. Basic Create Retrieve Delete & Update requests and response messages are used to develop application. Different modules can be developed using these Test methods and results. Remote Procedure Calls for Authentication and security could be one of them. In the future work of this project, there are many open source sub-projects of IoTDM middleware and application development where anyone can contribute.

Our oneM2M IoTDM handles Things/Devices that interact via oneM2M formatted messages over the supported wire protocols HTTP & CoAP. But if our things are not onM2M complaint then some code has to be written for custom plug-ins to handle those custom protocol messages. Therefore a lot of work needs to be done in the IoT southbound wire-protocols bindings to support as many protocols as possible.

RESTconf architecture is the simplest and most convenient way to develop applications for IoTDM but it is a non-oneM2M standard method for accessing the tree. But for the implementation of IoTDM it doesn't make any difference, instead the simple and extensible protocol like this is a key for the development of IoT applications as many developers prefer it over other methods.

Further, Its easy to standardize but very difficult to make everyone to comply to those standards. A push to create a Internet of Things will fail unless electronics firm collaborate more. All the businesses has their own custom systems for IoT solutions and if they comply to these common new standards then a significant amount of effort is required to make it compatible with their legacy network and this willingness is currently absent. Also developers are doing their best to implement IoTDM as closely to the oneM2M specifications as possible but the fact that these specifications are still evolving makes it further difficult for the vendors to implement. Slowly but surely we are progressing towards a common IoT standards which will make the vast deployment more easy.

6 References:

- https://en.wikipedia.org/wiki/OneM2M
- http://www.onem2m.org/
- https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#6a5ee4111d09
- https://en.wikipedia.org/wiki/Internet_of_things
- http://www.onem2m.org/technical/published-documents
- Paper: Research on Unified Data Model and Framework to Support Interoperability between IoT Applications by Kim, Choi & Hong
- https://wiki.opendaylight.org/view/IoTDM:Main
- Functional Architecture V2.10.0 TS0001
- Service Layer Core Protocol V2.7.1 TS0004
- https://join-iotdm.herokuapp.com/
- Software Defined Networking: A Comprehensive Survey
- https://www.getpostman.com/
- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://cisco.app.box.com/s/9ca2fvnqablirbsq5jwtsvxd6s4elkxw
- http://sdntutorials.com/what-is-restconf/
- oneM2M Webinars material : http://www.onem2m.org/insights/webinars