Internet Engineering Task Force
INTERNET-DRAFT
Intended Status: Standard Track
Expires: April 19, 2016

Vasu K
Rahul A J
Yangneng
Huawei

Oct 19, 2015

Service Provisioning for Constrained Devices
draft-vasu-core-ace-service-provisioning-00

Abstract

As more constrained devices are integrating with current Internet, the ubiquitous computing in scenarios like smart home is very important. In smart home, the constrained devices (ex. thermostat) need to be provisioned in such a way that it can inter-operate with any kind of devices like other constrained devices (ex. Air conditioner) or client devices (ex. smart phone). This document provides a method to support service provisioning based on pre-configured admission and resource control policies, where this method explains device’s service access in two different use cases: first provisioning the service when a constrained device accessing the service provided by other constrained device, second, accessing the service provided by constrained device from the client device (non constrained device).

Table of Contents

1 Introduction 3
2 Motivation 5
3 Terminology 5
4 System Architecture 6
5 Network Topology 9
6 Operations 10
6.1 Register Service 10
6.3.1 Resource Control 14
6.4 Search for services by device 18
6.5 Service request and response 18
7 Security Considerations 22
8 IANA Considerations 22
9 References 23
9.1 Normative References 23
9.2 Informative References 23
10 Acknowledgements 24
Authors' Addresses 24

1 Introduction

The work on Constrained Restful Environment (CoRE) aimed to realize the restful architecture for constrained devices [RFC7228] in constrained networks [RFC4944]. The CORE work group has recently standardized constrained application protocol (CoAP) [RFC7252] for interacting with constrained resources where general HTTP is not memory/energy efficient. The use of web linking for resources description and discovery hosted by constrained web servers is specified by CORE [RFC6690]. Even though, CoAP allows the direct resource access for constrained devices, it is not advisable for direct access of resources in networks where multicast procedures are infeasible due to heavy network load, and the networks where sleepy nodes exist. So, the CoRE working group comes up with a solution called resource directory (RD) [draft-ietf-core-resource-directory] to host the devices service information, and allow other devices to perform lookup procedures through .well-known/core path to resources.

The services advertised by these constrained devices needs to be commissioned and provisioned properly to allow other devices to access it. CoRE RD solution is a directory based solution that depends on CoAP protocol. CORE RD solution uses registration/update/delete/lookup procedures for service registration, service update, deleting service, lookup of services respectively. Service commissioning is a method which verifies a pre registered services with special commissioning tools/agents. These tools can be tablets or special embedded devices which initially stores the devices identifications in secure manner. Once the services are advertised by any device, those services need to be verified using commissioner. CORE RD provides a standard procedure to interact with commissioner, where commissioner acts like a client device to look up and verify the advertised services. Once the commissioner verifies the pre-registered services, commissioner can put some policy rules on services hosted by devices for resource control. These rules defined on (1) how to access the services either with other constrained devices or client devices, and (2) on operational instructions.

Architecture is defined to authenticate and authorize client requests for a resource on a server using logical entities such as client(C), client authorization manager(CAM), server(S), and server authorization manager(SAM)[draft-gerdes-ace-actors]. The main goal of delegated CoAP authentication and authorization framework (DCAF) is the setup of a datagram transport layer security channel between two nodes to securely transmit authorization tickets [draft-gerdes-core-dcaf-authorize]. The CAM sends an access request message on behalf of client by embedding requested permissions in client authorization information (CAI) field of access request message to

SAM. A ticket grant message is sent from SAM by embedding the permissions given from the server on a specific resource in server authorization information (SAI) field of ticket grant message to the client. These SAI, CAI use authorization information format (AIF) that describes the permissions requested from access request in a ticket request, where the underlying access control model will be that of an access matrix, which gives a set of permissions for each possible combination of a subject and an object [draft-bormann-core-ace-aif]. This simple information model also doesn’t allow conditional access (e.g.,"resource /s/tempC is accessible only if client belongs to group1 and does not belong to group2"). Finally, the model does not provide any dynamic functions such as enabling special access for a set of resources that are specific to a subject. But, the services provided by resources in constrained environment, need to be authorized and controlled conditionally based on some service level agreements or preconfigured policies on resource control.

Considering an example use case scenario such as thermostat device measures the current room temperature, and can service for air conditioner device to set automatic temperatures. In a smart home, user wants to regulate his room temperature automatically using his airconditioner device. Here, this airconditioner device can adjust its temperature to either cool the room or heat the room by accessing the service provided by the thermostat. Suppose this user leaves the home in the morning in hot summer and leaves the office in the evening to reach to home. But, before he reaches his room he wants to make his room cool enough. So he has to switch on the airconditioner from his mobile one hour before he leaves the office. So, before adjusting his aircconditioner to make the room cool enough, he might have to know the current room temperature. Thus he access the service provided by the thermostat to read the room temperature and adjust the airconditioner. However, there is a problem here on how to access these services which are provided by user’s home devices itself, what is the authenticity level to access from outside the home, even within home what is the access control/resource control of these devices because the neighboring device which are not authenticated can also access these service if those devices are within the constrained network range. Finally it is important to admit access of the service by client based on the configuration policies so that the devices can be protected from hazardous conditions, and allows only pre-agreed operations on devices.

The service provisioning presented in this document provides a method to support admission, and resource control policies using commissioning procedure. The method explains the device’s service access in two different use cases: first provisioning the service when a constrained device accessing the service provided by other constrained device, second, accessing the service provided by constrained device from the client device. Even though it is out of scope of the present document, it also considers a secure way of service commissioning as part of security.

2 Motivation

CORE RD solution provides various automated operations such as service registrations, service update, service removal, and service lookups initiated by endpoints and clients. However, managing this centralized directory server by allowing authorized users to perform these tasks, setting some service level agreements on clients to access these services, and providing limited or scope oriented lookups by other endpoints or clients require efficient service provisioning mechanism. The service provisioning method presented in this document deals on how a registered service from devices can be accessed by various clients or other devices. Moreover, it also provides a method for handling this resource/service access control mechanism using web service model for efficient service provisioning from outside the constrained home environment.

3 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

o "CORE", CORE is a Constrained RESTful Environment providing a framework for resource-oriented application intended to run on constrained networks [RFC7228].

o "COAP" The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and networks [RFC7252].

o "RD" The Resource Directory (RD) is a directory based server to host the descriptions of resources and allowing the lookups to be performed for those resources by various client devices.

o "Commissioner" Commissioning agent is tool/device that verifies the devices operation, integrity check with the network.

o "Constrained Device" These are embedded computing devices that are expected to be as resource constrained in terms of RAM/ROM size, and to be deployed with the constrained environment such as 6LoWPAN Networks.

o "Client" A client device is like resource constrained client such as other constrained device (ex. Air conditioner) or rich client devices such as Mobile/Laptop/Tablet etc, which access the services hosted by constrained devices (ex. thermostat).

o "Provisioning Server" this server is a process of verifying service requester, providing access controls or admission controls on resources to be accessed and inter-operating with various devices without bothering about kind of network protocols used. It also provides web access model outside the constrained environment.

o "Device Profile" A device profile comprises a set of attributes that are associated with a particular device. These include services, features, names, descriptions etc.

4 System Architecture

The system architecture is better explained with two different scenarios: (1) Constrained device access the service advertised by other constrained device is as shown in Fig 1. Here, one constrained device such as air-conditioner can access the service such as current room temperature advertised by other constrained device (ex. thermostat). This advertised service is to be commissioned by commissioner, and then it should be set with some admission and resource control policies by provisioning server. And, finally the service is allowed to advertise its service access from other constrained devices. Any device that is interested in that advertised service, need to do service lookup from RD Server. Once obtaining the path to the advertised service, the constrained client device can request a service to the device which hosts the service. Before sending the request, it MUST establish a secure channel between these two nodes [draft-schmitt-ace-twowayauth-for-iot]. Once the incoming request comes from the constrained client device, the device which hosts the service MUST authorize and provision for conditional access of its service from the provisioning server. The notification regarding the registered services to the commissioning agent can be sent from the RD server, which can be implementation specific and left for the user to choose any standard procedures and is out of scope of present document. Detailed operational procedure will be explained in the later sections of this document.

```
       +------+  +-------+   +-----+   +--------+  +----------+
       |Device1  |Device2|   | RD  |   |Provis  |  |Commision
       |(Air     |(Thermo|   |     |   |ioning  |  |ing        |
       |Condi |  |  stat)|   |Serv     |Sever   |  |Agent      |
       |tioner)  |       |   |er   |   |        |  |           |
       +--|---+  +----|--+   +--|--+   +----|---+  +-----|----+
          |           |         |           |            |
          |           |         |           |            |
          |           |Register |           |            |
          |           ----------//          |            |
          |           | Service/|  Verify Preregistered\ |/
          |           |         ------------------------//
          |           |         |    Service|         // |
          |           |         |           |        /   |
          |           |         |           |            |
          |           |         |           |            |
          |           |         |           |            |
          |           |         |           |//Define    |
          |           |         |          /-------------
          |           |         |         / \ Policies   |
          |Search a Service  \  |           |            |
          ---------------------//           |            |
          |           |       //|           |            |
          |           |      /  |           |            |
          |           |         |           |            |
          |           |         |           |            |
          |Request    |         |           |            |
          -----------/          |           |            |
          |Service   /|         |           |            |
          |         / |         |           |            |
          |           |Check for authorization           |
          |           |admission, Resource  |            |
          |           ---------------------//            |
          |           | Control Policies  //|            |
          |           |         |        /  |            |
          |           |         |           |            |
          |           |         |           |            |
          |           | // Service Grant/Deny            |
          |           /---------------------             |
          |          /|\        |           |            |
          |  /        |         |           |            |
         \//Service  |          |           |            |
          /\----------          |           |            |
          |    Grant/Deny       |           |            |
          |           |         |           |            |
```

      Fig 1.Constrained device accessing service from constrained device

```
       +--------+  +-------+  +-------+  +---------+ +---------+
       |Client  |  |Device2|  |RD     |  |Provision| |Commissi |
       |(Smart  |  |(Thermo|  |Server |  |ing Server |oning    |
       | Phone) |  | stat) |  |       |  |         | |Agent    |
       |        |  |       |  |       |  |         | |         |
       |        |  |       |  |       |  |         | |         |
       +---|----+  +---|---+  +---|---+  +----|----+ +-----|---+
           |           |          |           |            |
           |           |          |           |            |
           |           |Register  |           |            |
           |           -----------/           |            |
           |           |Service   /           |            |
           |           |         /|           |            |
           |           |          |  /        |            |
           |           |          |//Verify Preregistered  |
           |           |          --------------------------
           |           |          |\    Service            |
           |           |          |           |            |
           |           |          |           |            |
           |           |          |           |  /         |
           |           |          |           |//Define    |
           |           |          |           -------------
           |           |          |           |\ Policies  |
           |           |          |           |            |
           | Request for Service  |        \               |
           ----------------------------------//            |
           |           |          |         //|            |
           |           |          |        /  |            |
           |           |          |           |            |
           |           |  /       |           |            |
           |           |// Request| for Service            |
           |           -----------------------             |
           |           |          |           |            |
           |           |          |           |            |
           |           |          |           |            |
           |           | Service Grant/Deny\  |            |
           |           ----------------------/             |
           |           |          |         //|            |
           |           |          |        /  |            |
           |           |          |           |            |
           |           |          |           |            |
           |  //       |          |           |            |
           |//   Service Grant/Deny           |            |
           \----------------------------------            |
           | \         |          |           |            |
           |           |          |           |            |
```

Fig 2. Client accessing service from Constrained device

2) Client device access the service advertised by constrained device is as shown in Fig 2. For example, the client device such as smart phone can access the service (ex. room temperature) advertised by other constrained device (ex. thermostat). The client can access the service within a home environment or outside the home environment. So, in this scenario, the provisioning server maintains the service as a web service.

This advertised service is to be commissioned by commissioner, then to be set with some admission and resource control policies by provisioning server. And, finally the service is allowed to advertise its access from the client devices. Any client that wishes to access this web service looks for corresponding operations provided from the provisioning server.

5 Network Topology

The constrained devices such as Thermostat, Airconditioner may use small memory constrained sensors/actuators for simple services such as cooling/heating the room or just to measure the current room temperature. These memory constrained embedded devices may implement the 6LoWPAN stack such as uIP (provided by Contiki), and provide access for communication to other external queries from client devices such as smart phone which typically implements rich stack TCP/IP. Even though RD server or Provisioning server are shown as separate servers in the LAN as given in Fig 3, these can be hosted on a single server running two different processes. Moreover, the commissioner implements a standard procedure to interact with devices as a separate agent process which is out of scope of the present document and has been left to user’s choice while satisfying the mentioned operations in the current draft. On the other hand, these specific operations can be implemented separately as a third party and to be used at the commissioning agent. The lower level communication technology can be implemented either through Bluetooth (BT) or near field communication (NFC) to verify the devices unique ID (for ex. using MAC). Even though, the implementation procedure for commissioner is out of scope for the present document, it is shown as sample interaction with RD server/provisioning server as part of commissioning procedure in subsequent sections. Even though the present document discusses about 6LoWPAN based sensor network, it can be easily moved to any other technology such as Zigbee/BLE/Wireless HART without any changes in the architecture or design, because the present document abstracted the communication networks with their edge routers. The communication and routing mechanisms or procedure between edge router and sensor devices/client devices are out of scope of the present document.

```
             -------                      --------
           //       \                  //        \
          /                          //            \
         /                          /
        /                          /
        | +--------+    |           |        +--------+|
       |  |RD Server---| |         | +-----+ |Thermostat|
       |  +--------+   | | LAN     | |Edge | +--------+ |
       |               |------|-------     |            |
       |  +----------+ | |    |    | |Router 6LoWPAN    |
        | |Provision --||     |     |+-----+           |
         |Server    |  /     |          +--------+   /
         +----------+ /      |          |Aircondioner
                     /       |       \  +--------+//
           \       //        |         \        //
             -------          |           --------
                              |
                              |
                              |
                              |
                              |
                              |
                             -|-----
                          //- |     -\
                        //  +-|----+   \
                       /    |Edge  |
                      /     |Router|
                      |     +------+      |
                     |                     |
                     |    WiFi             |
                     |   +-------+ +-----+ |
                     |   |Smart  | |Commisioning
                      |  |Phone  | | Agent|
                        +-------+ +-----+/
                                        /
                        \             //
                          \-       -//
                             -------
```

Fig 3. Network Topology

6 Operations

6.1 Register Service

The constrained device which hosts the service MUST register its service with the RD server using its unique identifier (for ex. MAC id, UDDI registry etc.) and IP address as shown in Fig 4. The device MUST send a POST request for registering its service.

Before sending a request, it MUST establish a secure channel between these two nodes [draft-schmitt-ace-twowayauth-for-iot]. Once the service has been registered with the RD server, the RD server may notify the registered information of a device (for ex.its unique identifier and device name) to a commissioning agent.

```
   +---------+                                       +---------+
   |         |                                       |         |
   | Device  |                                       |RD Server|
   |         |                                       |         |
   +----+----+                                       +-----+---+
        |                                                  |
        |                                                  |
        |                                              ‘.  |
        | POST /rd?ep=node1&d=example.com&et=temperature-no‘.|
        +-------------------------------------------------,’.
        | gp=thermostat&con=DeviceID(100)                ,’  |
        |                                              ,’    |
        |   ,’                                               |
        | ,’  2.01 Created Location: /rd/7521                |
        ‘.-------------------------------------------------
        | ‘-.                                              |
        |    ‘.                                            |
        |                                                  |
```

Fig. 4 Registering a Service

6.2 Verify pre-registered service

The commissioning agent MUST verify any pre registered service with the RD server as shown in Fig 5. The commissioning agent sends a GET request for domain lookup. Before sending the request, it MUST establish a secure channel between these two nodes [DTLS][TLS]. Once obtaining the specific domain, it MUST look for the group to which the service belongs. Once obtaining the specific domain and group, it MUST send a service look up with the RD server for the registered service. Once obtaining the service information about a specific device, the commissioning agent MUST verify the registered service. This service information is later used to create service registry in the provisioning server as explained in the following section. The example service information (denoted as SRV) looks like as shown in Fig 6.

```
       +---------------+                                  +----------+
       |Commissioning  |                                  | RD Server|
       |Agent          |                                  |          |
       +------+--------+                                  +--------+-+
              |                                                    |
              |                                                    |
              |     GET /rd-lookup/d                           '.  |
              +-----------------------------------------------------:'.
              |                                                .'  |
              |                                                    |
              | .'2.05 Content </rd>;d=example.com,</rd>;d=example.com
              ::---------------------------------------------------+
              | '-.                                                |
              |   GET /rd-lookup/gp?ep=node1&d=example.com      '. |
              +------------------------------------------------------/.
              |                                                 .' |
              |                                                    |
              | .'2.05 Content <coap://ip:port>;gp=thermostat;ep=node1
              ::---------------------------------------------------+
              | '-.                                                |
              |                                               '.   |
              | GET /rd-lookup/res?rt=temp&gp=thermostat&d=example.com
              +-----------------------------------------------------:'.
              |                                                .'  |
              |                                                    |
              | .'2.05 Content <coap://host:port>;rt=temp;gp=thermostat
              ::---------------------------------------------------+
              | '-.                                 d=example.com      |
      +----------------------------+                               |
      |Authentication of Service   |                               |
      |Info and DeviceID           |                               |
      +----------------------------+ POST Verified User; DeviceID'. |
              +-----------------------------------------------------::
              | .'                                              .-' |
              .'.___________________________________________________|
              |  '.    2.00 OK                                     |
```

Fig. 5 Verify pre registered service

```
              SRV {
                    Name: Node1
                    Group: Thermostat
                    Domain: myhome.com
                    Type: Temperature node
                    Device ID: 1001
                    Device IP: <host:port>
                  }
```

Fig 6. Example Service Informaion

6.3 Define policies on resource control

```
 +----------------+                        +---------------+
 |Commissioning   |                        |Provisioning   |
 | Agent          |                        | Server        |
 +------+---------+                        +---------+------+
        |  POST /thermostat /HTTP/1.1            '. |
        +-----------------------------------------/.
        |  HOST thermostat.ps.example.com        .' |
        |  Content-Type: application/text           |
        |  SRV { Name: Node1                        |
        |        Group: Thermostat                  |
        |        Domain: myhome.com                 |
        |        Type: Temperature-node             |
        |        DeviceID: 1001                     |
        |        DeviceIP: <host:port> }            |
        |                                           |
        | .'     HTTP/1.1 200 OK                    |
        ::-----------------------------------------'
        | '.     Content-Type: application/text     |
        |        { sID (service ID) }               |
        |                                           |
        |  POST /thermostat /HTTP/1.1            '. |
        +-----------------------------------------::
        |  HOST thermostat.ps.example.com        .' |
        |  Content-Type: application/text           |
        |  AC { ServiceID: 1234                     |
        |       Auth: Basic Auth Support            |
        |       Count: 10                           |
        |       Admission Control: R,W,R/W,D }      |
        |                                           |
        | .'    HTTP/1.1 200 OK                     |
        ::-----------------------------------------+
        | '.    Content-Type: application/text      |
        |                                           |
        |  POST /thermostat /HTTP/1.1            '. |
        +-----------------------------------------::
        |  HOST thermostat.ps.example.com        .' |
        |  Content-Type: application/text           |
        |  RC { If C is from G1 allow {R,W};        |
        |       If C is from G2&!G3 allow {R};      |
        |       If C is from d1&g1 allow {R,W,D};   |
        |        : }                                |
        | .'    HTTP/1.1 200 OK                     |
        ::-----------------------------------------+
        | '.    Content-Type: application/text      |
        |                                           |
```

Fig. 7 Defining Policies on Resource and Access Control

Once the hosted service has been verified by commissioning agent (CA), the CA MUST create a service registry with the provisioning server as explained in Fig 7. The provisioning server SHOULD send a service ID as a response back to the commissioning agent after creating the service entry.

This service ID can be later used by the commissioning agent to permanently DELETE the service entry (if required). The commissioning agent MUST create some admission control policies such as read (R), write (W), read/write (R/W), delete (D), number of simultaneous connection on resource etc. on the registered service. Once the admission control policies has been set on a specific device, the resource control policies such as conditional access of a service, quality of service agreements (based on the priority levels set for clients) can be set on that registered service. These conditional access on service can be implemented with simple conditional statements as explained in section 6.3.1 (for ex. "client (c) can access service with only read (R), write (W) permissions if it only belongs to group (g)"). The implementation or information format details of these conditional statements is out of scope of the present document (TBD). The example admission control and resource control policies are as shown in Fig 8, and Fig 9 respectively.

```
AC {
      Service ID: 12345
      Auth: Basic Auth Support
      Count: 10
      Admission Control: R, W, R/W, D
        :
        :
   }
```

Fig 8. Example Admission Control Policies

```
RC {
      If c is from g1 allow {R,W}
      If C is from g2 & !g3 {R}
      If C is from d1 & g1 allow {R, W, D}
        :
        :
   }
```

Fig 9. Example Resource Control Policies

6.3.1 Resource Control

Resource control policies for constrained devices are expressed in terms of conditional expressions as explained in Fig. 9. Consider a scenario where we define the client (C) (who accesses the resource) in terms of groups/levels. For example in a typical home building, we assign each floor as a group. Suppose for a three floor building, the clients such as mobile phone/air conditioner can belong to any of the floor within a building. And we allow various permissions for the clients according to the group it belongs to, as specified in Fig 10.

Client	R	W	U	D
G1	*	-	*	-
G2	*	*	-	-
G3	-	-	-	*

Fig 10. Example Permissions on Methods

Supposed we assigned the priorities for different groups as C belongs to {G1, G2, G3} => {P1, P3, P2}. Moreover, if we would like to assign different QoS classes for clients, depending on the applications they use then it is required to control QoS policies in resource control. QoS is defined in terms of various parameters such as {availability, reliability, serviceability, data accuracy, aggregation delay, coverage, fault tolerance, network lifetime} in wireless sensor networks. It is assumed that based on these parameters, QoS is defined in terms of various classes such as {Q1, Q2, Q3}, then it is required that some of the clients can make some pre-level agreements on QoS requirement for their applications either based on the groups it belongs to or based on the priority of the clients request (Suppose, C belongs to {Q1, Q2, Q3}). Method for defining QoS classes is out of scope of the present document. Once defining the groups, its priorities, QoS classes, and permissions, then the conditional statements which define the resource control policies can be defined as follows:

ST1: If the client belongs to G1 then it is allowed with permissions {R, R/W, U}, priority {P1}, QoS {Q1}, and operations {turn it up, read}; else if the client belongs to G2 then it is allowed with permissions {R, W, R/W}, priority {P3}, QoS {Q2}, and operations {turn it up, read}; else if the client belongs to G3 then it is allowed with permissions {D}, priority {P2}, QoS {Q3}, and operations {turn it down}.

ST2: Allow the client with priority {P1}, QoS {Q1}, operations {turn it up, turn it down, read}, and allow only with permissions {R} in G1; permissions {R, R/W, D} in G2; and permissions {D} in G3.

ST3: Allow the client with priority {P1}, QoS {Q1}, and allow with permissions {R}, operations {read} in G1; allow with permissions {R, R/W, D}, operations {turn it up, turn it down, read} in G2; and allow with permissions {D}, operations {turn it down} in G3.

Above conditional statements are few examples on how to define the conditional statements, the statements can be defined on any manner based on the resource control policies we would like to achieve. The above statements can be better explained in plain semantic notation as shown in Fig 11(a)-13(a), and the corresponding JSON representations for message exchange is explained in Fig 11(b)-13(b). These statements can be even implemented using data modeling language such as YANG or ASN 1.1 which is out of scope of the present document.

```
 C
 {
    G1                                 |"[
    {                                  |"C":{"G1":{"Allow":"R,U",
        Allow {R,U}                    |"Priority":"P1","QoS":"Q1",
        Priority {P1}                  |"Operations":"turnup,read"},
        QoS {Q1}                       |"G2":{"Allow":"R,W",
        Operations {tunr it up, read}|"Priority":"P3","QoS":"Q2",
    }                                  |"Operations":"turn it
    G2                                 |up,read"},"G3":{"Allow":"D",
    {                                  |"Priority":"P2","QoS":"Q3",
        Allow {R,W}                    |"Operations":"turn it down"
        Priority {P3}                  |}}]"
        QoS {Q2}                       |
        Operations {turn it up, read}|
    }                                  |
    G3                                 |
    {                                  |
        Allow {D}                      |
        Priority {P2}                  |
        QoS {Q2}                       |
        Operations {turn it down}    |
    }                                  |
 }                                     |

               (a)                                    (b)
```

Fig 11. ST1: (a) Semantic Notation (b) JSON Representation

```
       C                                   | "[
       {                                   | "Priority":"P1","QoS":"Q1",
          Priority {P1}                    | "Operations":"turn it up,
          QoS {Q1}                         | turn it down, read",
          Operations {turn it up,turn it | "C":{"G1":{"Allow":"R"},
                      down, read}         |      "G2":{"Allow":"R,W,D"},
          G1                               |      "G3":{"Allow":"D"}}
          {                                | ]"
              Allow {R}                    |
          };                               |
          G2                               |
          {                                |
              Allow {R,W,D}                |
          };                               |
          G3                               |
          {                                |
              Allow {D}                    |
          };                               |
       }                                   |

                    (a)                                 (b)
            Fig 12. ST2: (a) Semantic Notation  (b) JSON Representation

       C                                      | "[
       {                                      | "Priority":"P1","QoS":
           Priority {P1}                      | "Q1","C":{"G1": {"Allow":
           QoS {Q1}                           | "R","Operations":"read"},
           G1                                 | "G2":{"Allow":"R,W,D",
           {                                  | "Operations":"turn it up,
               Allow {R}                      | turn it down, read"},
               Operations {read}              | "G3":{"Allow":"D",
           };                                 | "Operations":"turn it
           G2                                 | down"}}]"
           {                                  |
               Allow {R,W,D}                  |
               Operations {turn it up, turn |
                           down, read}        |
           };                                 |
           G3                                 |
           {                                  |
               Allow {D}                      |
               Operations {turn it down}      |
           };                                 |
       }                                      |

                    (a)                                  (b)
            Fig 13. ST3: (a) Semantic Notation (b) JSON Representation
```

6.4 Search for services by device

Any client device (as explained for scenario 2) MUST interacts with the provisioning server and looks for deployed services by devices. Moreover, the provisioning server can verify the complete authorization, admission, and resource control of any device’s services. Whereas, if any other constrained devices (ex. air conditioner) searches for services hosted by other constrained device (as explained for scenario 1) MUST interact with the RD server as shown in Fig 10. Here, initially the device queries for all services that are hosted by other devices, then it searches within the domain for specific service, its SRV info, and path to the hosted service. Before sending a request, it MUST establish a secure channel between these two nodes [draft-schmitt-ace-twowayauth-for-iot].

```
 +---------------+                                   +----------+
 | Device        |                                   | RD Server|
 | (aircondit    |                                   |          |
 |   ioner)      |                                   |          |
 +-----+---------+                                   +-------+--+
       |                                                     |
       |  GET /rd-lookup/gp?d=example.com                ‘.  |
       +---------------------------------------------------‘.:
       |                                                 .-’ |
       | .’2.05 Content <gp="thermostat">                    |
       ::----------------------------------------------------+
       | ‘-.                                                 |
       |   GET /rd-lookup/ep?gp=thermostat                ‘. |
       +----------------------------------------------------::
       |                                                  .’ |
       | .’2.05 Content <Node1> <Node2>                      |
       ::----------------------------------------------------+
       | ‘.                                                  |
       |                                                     |
       | GET /rd-lookup/ep?et=temperature&gp=thermostat   ‘. |
       +----------------------------------------------------‘.
       |                                                  .’ |
       |                                                     |
       | .’2.05 Content <coap://ip:port>;ep="Node1"          |
       ::----------------------------------------------------+
       | ‘-.                                                 |
       |                                                     |
```

Fig. 10 Search for services by device

6.5 Service request and response

In scenario 1 (as shown in Fig 1), service request and response MUST use coap based communication to access the service as shown in Fig 11. Before sending a request, it MUST establish a secure channel between these two nodes [draft-schmitt-ace-twowayauth-for-iot]. Suppose, the constrained client device (for ex. airconditioner) want to access the service hosted by another constrained device (for ex. thermostat), then the client device MUST send a coap based GET request to thermostat. Then, this device (thermostat) SHOULD send a POST request to provision this service request with the provisioning server by sending clients <IP:port>. Based on the clients <IP:port>, the provisioning server MUST find the client (ex. airconditioner) details such as service information, group, domain, and type details.

```
+------------+             +-------------+         +-----------+
|Airconditi  |             |Thermostat   |         |Provision  |
|oner        |             | (IP1)       |         |ining Server
| (IP2)      |             |             |         | (IP3)     |
+-----+------+             +------+------+         +--------+--+
      |                           |                         |
      |coap://thermostat.        '.|                         |
      +---------------------------::                        |
      |    example.com/temp     .' |POST /thermostat       '. |
      |                           +-------------------------::
      |                           |HOST thermostat.ps.    .' |
      |                           |             example.com  |
      |                           |Content-Type: application/txt
      |                           |{ SRC: <IP1,port>         |
      |                           |  DST: <IP3,port>         |
      |                           |  Client: <IP2,port> }    |
      |                           |                          |
      |                           |  +-------------------------+
      |                           |  |Check for Admission,     |
      |                           |  |ResourceControl of thermost
      |                           |  |for airconditioner       |
      |                           |  +-------------------------+
      |                           |                          |
      |                           | .'2.00 OK { Permit/Deny }|
      | .'URI-Path: temp CON 200    ::------------------------+
      ::-------------------------+ '-.                       |
      | '.("thermostat","aaaa::212.|                         |
      |  7402.2.202","temp",27)    |                         |
      |                           |                          |
```

Fig. 11 Request/Response within Constrained Environment

Once the client is identified, the provisioning server MUST check for authorization, admission and resource control policies of hosted service (ex. thermostat). Once the service request is authorized to access then the URI-Path for hosted service along with the value is sent as a coap response to client device (air conditioner). Here, the request is conditional i.e. based on the resource control policies of a resource (such as thermostat) for a client (airconditioner), the permissions are given to access the resource.

```
+-------------+               +------------+           +---------+
|             |               |Provision   |           |         |
| Client      |               |ining Server|           |Thermostat
|             |               |            |           |         |
+-----+-------+               +-----+------+           +------+--+
      |                             |                         |
      |http://thermostat.        '. |                         |
      +----------------------------::                         |
      |  example.com/temp        .' |                         |
      |            +-----------------------------+            |
      |            |Check for Admission,         |            |
      |            |Resource Control of thermostat            |
      |            |for airconditioner           |            |
      |            +-----------------------------+            |
      |                             |                         |
      |                             | coap://thermostat.   '. |
      |                             +-----------------------::
      |                             |example.com/temp      .' |
      |                             |                         |
      |                             |                         |
      |                             | .'URI-Path: temp CON 200|
      |                             ::------------------------+
      |                             | '.                      |
      | .' HTTP/1.1 200 OK          |                         |
      ::----------------------------+                         |
      | '. Temperature: 27          |                         |
      |                             |                         |
```

Fig. 12 Request/Response from outside Constrained Environment

Service request and response in scenario 2 (as shown in Fig 2), uses simple http based communication to access the service from the PS. Provisioning Server then sends a coap based GET request to the ultimate device that hosts service. Before sending this request to the actual device for service, PS authorizes the service request. Once, the service request is authorized to access, then the URI-path for hosted service along with the value is sent as HTTP response to client device. PS can implement a reverse proxy case for HTTP-CoAP protocol translation defined in [draft-ietf-core-http-mapping].

```
------------------HTTP begin ------------------------------------
HTTP POST
Request:
POST /thermostat   /HTTP/1.1
HOST thermostat.example.com
Content-Type:  application/x-www-form-urlencoded
Content-Length:  length
licenseID=string & content=string & paramsXML=string

Response:
HTTP/1.1   200 OK
Content-Type:  text/xml; charset=utf-8
Content-Length:  length
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://xyz.com/">
string
</string>

------------------HTTP end  ------------------------------------

------------------ REST via HTTP begin --------------------------
REST via HTTP POST
Request:
POST /thermostat   /HTTP/1.1
HOST thermostat.example.com
Content-Type:  application/x-www-form-urlencoded
Content-Length:  length

licenseID=string & content=string & paramsXML=string

Response:
HTTP/1.1   200 OK
Content-Type:  text/xml; charset=utf-8
Content-Length:  length

string

------------------REST via HTTP end -----------------------------

------------------SOAP begin ------------------------------------

SOAP 1.2
Request:
POST /Thermostat   /HTTP/1.1
HOST: www.example.org
```

```
Content-Type:  application/soap+xml; charset=utf-8
Content-Length:  length

<?xml version="1.0"?>
<soap:envelop>
Xmlns:soap=http://www.w3.org/2001/12/soap-envelop
Soap:encodingStyle=http://www.w3.org/2001/12/soapencoding>
<soap:body xmlns: m="http://www.myhome.org/thermostat">
<m:GetTemperature>
<m:thermostat>1</m:thermostat>
</m:GetTemperature>
</soap:body>
</soap:envelop>

Response:
HTTP/1.1 200 OK
Content-Type:  application/soap+xml; charset=utf-8
Content-Length:  length

<?xml version="1.0"?>
<soap:envelop>
Xmlns:soap=http://www.w3.org/2001/12/soap-envelop
Soap:encodingStyle=http://www.w3.org/2001/12/soapencoding>
<soap:body xmlns: m="http://www.example.org/thermostat">
<m:GetTemperatureResponse>
<m:temperature>27.8</m:temperature>
</m:GetTemperatureResponse>
</soap:body>
</soap:envelop>

------------------SOAP end ----------------------------------
```

7 Security Considerations

Security level for message authentication is out of scope of the present document. However, the following security consideration needs to be considered for the present proposed method. Services that run over UDP are unprotected and vulnerable to unknowingly become part of a DDoS attack as UDP does not require return routability check. Therefore, an attacker can easily spoof the source IP of the target entity and send requests to such a service which would then respond to the target entity. The TLS/DTLS based security solution can be considered for secure message communication.

8 IANA Considerations

TBD

9 References

9.1 Normative References

9.2 Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.

[draft-ietf-core-resource-directory] Shelby, Z., and Bormann, C., "CoRE Resource Directory", draft-ietf-core-resource-directory-02 (work in progress), November 2014.

[draft-gerdes-ace-actors] Gerdes, S., "Actors in the ACE Architecture", draft-gerdes-ace-actors-03 (work in progress), March 2015.

[draft-gerdes-ace-dcaf-authorize] Gerdes, S., Bergmann, O., Bormann, C., "Delegated CoAP Authentication and Authorization Framework (DCAF)", draft-gerdes-ace-dcaf-authorize-02, March 2015.

[draft-bormann-core-ace-aif] Bormann, C., "An Authorization Information Format (AIF) for ACE", draft-bormann-core-ace-aif-oo, January 2014.

[draft-schmitt-ace-twowayauth-for-iot] Schmitt, C., Stiller, B., "Two-way Authentication for IoT", draft-schmitt-ace- twowayauth-for-iot-01, December 2014.

[DTLS] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 6347, January 2012.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.2", RFC 5246, August 2008.

[draft-ietf-core-http-mapping] Castellani, A., Loreto, S., Rahman, A., Fossati, T., and Dijk, E., "Guidelines for HTTP-CoAP Mapping Implementations", draft-ietf-core-http-mapping-05, (work in progress), Oct 2015.

10 Acknowledgements

Special thanks to Amit Kumar S,Zhengfei, Fubaicheng, Yangjun,Vijayachandran Mariappan, Shashidhar C Shekar, Jayaraghavendran K, Ajay Sankar, Puneet Balmukund Sharma, and Rabi Narayan Sahoo for extensive comments and contributions that improved the text.

Thanks to Hedanping (Ana), Behcet Sarikaya, and Carsten Bormann for helpful comments and discussions that have shaped the document.

Authors' Addresses

Vasu K
Huawei Technologies
Bangalore
India

EMail: vasu.kantubukta@huawei.com

Rahul A Jadhav
Huawei Technologies
Bangalore
India

EMail: rahul.jadhav@huawei.com

yangneng
Huawei Technologies
Shenzhen
China

EMail: yangneng@huawei.com