T2TRG Research Group
Internet-Draft
Intended status: Informational
Expires: September 10, 2020

J. Jimenez
Ericsson
March 09, 2020

Using MUD on CoAP environments
draft-jimenez-t2trg-mud-coap-00

Abstract

This document provides some guidelines on how to add Manufacturer Usage Descriptions (MUD) to CoAP environments.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Manufacturer Usage Descriptions (MUDs) have been specified in [RFC8520]. As the RFC states, the goal of MUD is to provide a means for end devices to signal to the network what sort of access and network functionality they require to properly function.

Schemes that rely on connectivity to bootstrap the network might be flaky if that connectivity is not present, potentially preventing the device from working correctly in the absence of Internet connectivity. Moreover, even in environments that do provide connectivity it is unclear how continued operation can occur when the manufacturer's server is no longer available.

While [RFC8520] contemplates the use of CoAP [RFC7252] in the form of CoAP URLs, it does not explain how MUDs can be used in a CoAP network. Moreover, in CoAP the MUD file can be hosted on the CoAP endpoint itself, instead of hosting it on a dedicated MUD File Server.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. MUD Architecture

MUDs are defined in [RFC8520] and are composed of:

o A URL that can be used to locate a description;

o the description itself, including how it is interpreted; and

o a means for local network management systems to retrieve the description;

o which is retrieved from a MUD File Server.

In a MUD scenario, the end device is a "Thing" that exposes a "MUD URL" to the network. Routers or Switches in the path that speak MUD can forward the URL to "MUD Managers" that query a "MUD file server" and retrieve the "MUD File" from it. After processing, the "MUD Manager" applies an access policy to the IoT Thing.

```
 .......................................                      +-------+
 .                      ____________   .                      |  MUD  |
 .                     +            +  .          +-----------+-+File |
 .                     |    MUD     +-->get URL+->+    MUD      +-----+
 .                     |  Manager   |  .(https)   | File Server |
 .  End system network +____________+<+MUD file<-<+-------------+
 .                             .       .
 .                             .       .
 . _______                 _________   .
 .+       + (DHCP et al.) + router  +  .
 .| Thing +---->MUD URL+->+   or    |  .
 .+_______+               | switch  |  .
 .                        +_________+  .
 .......................................
```

Figure 1: Current MUD Architecture

The general operation consists on a Thing emitting the MUD as a URL using DHCP, LLDP or through 802.1X, then a Switch or Router will send

the URI to some IoT Controlling Entity. That Entity will fetch the MUD file from a Server on the Internet over HTTP.

MUDs can be used to mitigate DDOS attacks to and from devices, for example by prohibiting unauthorized traffic to and from IoT devices. MUD also prevents devices from becoming the attacker, as that would require the device to send traffic to an unauthorized destination. Overall MUDs can be used to automatically permit the device to send and receive only the traffic it requires to perform its intended function.

This is indeed an important subject as trustworthy IoT operations is a recurring topic in the IETF [RFC8576].

2.1. Problems

The biggest issue with this architecture is that if the MUD File server is not available at a given time, no Thing can actually join the network. Relying on a single server is generally not a good idea. The DNS name may point to a load balancer group, but then that would require added infrastructure and complexity.

Another potential issue is that MUD files seem to be oriented to classes of devices and not specific device instances. It could be that during bootstrapping or provisioning different devices of the same class have different properties and thus different MUD files, more granularity would be preferable. This could be achieved by creating per-device MUD files on the server, but that mechanism does not seem to have been currently specified.

This brings us to the third problem, which is that the MUD file is somewhat static on a web server and out of the usual interaction patterns towards a device. In CoAP properties that are intrinsic to a device (e.g. sensing information) or configuration information (e.g. lwm2m objects used for management) are hosted by the device too, even if they could be replicated by a cloud server.

3. MUD architecture using CoAP

[RFC8520] allows a Thing to use the CoAP protocol scheme on the MUD URL. In this document we modify slightly the architecture. The components are:

- o A URL using the "coaps://" scheme that can be used to locate a description;

- o the description itself, including how it is interpreted, which is now hosted and linked from "/.well-known/core" and

- a means for local network management systems to retrieve the MUD description

- which is hosted by the Thing itself acting as CoAP MUD File Server.

```
            ...................................................
            .                                  ____________   .
            .                                 +            +  .
            .             +------------------ |    MUD     |  .
            .   get URL   |                   |  Manager   |  .
            .   (coaps)   |                   +____________+  .
            .  MUD file   |                         .         .
            .             |                         .         .
            .             |     End system network  .         .
            .             |                         .         .
            .           __v____                 _________     .
            .          +       + (DHCP et al.) + router  +    .
            .     +--- | Thing +---->MUD URL+->+   or    |    .
            .     |MUD +_______+               | switch  |    .
            .     |File  |                     +_________+    .
            .     +------+                                    .
            ...................................................
```

Figure 2: Self-hosted MUD Architecture

3.1. Problems

This design has some problems of its own. The main being that, if DHCP is restricted in the network, the device may not be able to advertise a functional MUD URL ([RFC8520] Section 1.9).

4. Finding a policy: MUD URL advertisement

A CoAP endpoint will emit a URL that uses the CoAP scheme [RFC7252]. This URL serves both to classify the device type and to provide a means to locate a policy file, as specified on [RFC8520].

4.1. Dynamic Host Configuration Protocol (DHCP)

[RFC8520] specifies a new MUD URL DHCP Option that carries the MUD URL to the DHCP Server. The functioning is the same as specified on [RFC8520].

If the DHCP Server cannot provide a valid IPv4 or IPv6 address, the device may use a temporary IPv6 link-local address as base URI (e.g. "coaps://[FE80::AB8]/mud/light-class"). As link local is not a

globally addressable address, the MUD Manager will only be able to reach the device if it is in the same network as the device.

TBD : IPv4, coap://, ...

4.2. Neighbor Discovery Protocol (NDP)

IPv6 hosts do not require DHCP to get access to the default gateway. [RFC4861] can also be used to advertise the MUD URL.

TBD : Figure out how these work - Neighbor Solicitation (type 135) - Neighbor Advertisement (type 136) - Redirect (type 137)

4.2.1. NDP on 6LoWPANs

LoWPANs are characterized as lossy, low-power, low-bit-rate, short-range; with many nodes saving energy with long sleep periods. For that reason vanilla NDP [RFC4861] might not be the most desiderable solution in such networks and optimizations like the ones provided by [RFC6775] are required.

TBD : Figure out how these work

4.3. Stateless autoconfiguration (SLAAC)

[RFC4862] specifies how to create and auto configure link-local addresses during system startup.

TBD : Figure out how these work

5. CoAP Operations

Things can expose MUDs as any other resource. MUD Managers can send a GET requests to a CoAP server for "/.well-known/core" and get in return a list of hypermedia links to other resources hosted in that server. Among those, it will get the path to the MUD file, for example "/mud" and Resource Types like "rt=mud".

5.1. Discovery

5.1.1. Resource Directory

By using [I-D.ietf-core-resource-directory], devices can register a MUD file on the Resource Directory and use it as a MUD repository too. Making it discoverable with the usual RD Lookup steps.

Lookup will use the resource type "rt=mud", the example in Link-Format [RFC6690] is:

```
REQ: GET coap://rd.company.com/rd-lookup/res?rt=mud

RES: 2.05 Content

     <coap://[2001:db8:3::101]/mud/box>;rt=mud;
       anchor="coap://[2001:db8:3::101]"
     <coap://[2001:db8:3::102]/mud/switch>;rt=mud;
       anchor="coap://[2001:db8:3::102]",
     <coap://[2001:db8:3::102]/mud/lock>;rt=mud;
       anchor="coap://[2001:db8:3::102]",
     <coap://[2001:db8:3::104]/mud/light>;rt=mud;
       anchor="coap://[2001:db8:3::104]"
```

The same example in CoRAL ([I-D.ietf-core-coral], [I-D.hartke-t2trg-coral-reef]) is:

```
REQ: GET coap://rd.company.com/rd-lookup/res?rt=mud
     Accept: TBD123456 (application/coral+cbor@identity)

RES: 2.05 Content
     Content-Format: TBD123456 (application/coral+cbor@identity)

     rd-item <coap://[2001:db8:3::101]/mud/box> { rt "mud" }
     rd-item <coap://[2001:db8:3::102]/mud/switch> { rt "mud" }
     rd-item <coap://[2001:db8:3::102]/mud/lock> { rt "mud" }
     rd-item <coap://[2001:db8:3::103]/mud/light> { rt "mud" }
```

5.1.2. Multicast

[RFC7252] registers one IPv4 and one IPv6 address each for the purpose of CoAP multicast. All CoAP Nodes can be addressed at "224.0.1.187" and at "FF0X::FD". Multicast could also be used to discover all Manufacturer descriptions in a subnet.

The example in Link-Format [RFC6690] is:

```
GET coap://[FF0X::FE]/.well-known/core?rt=mud
```

The same example in CoRAL ([I-D.ietf-core-coral], [I-D.hartke-t2trg-coral-reef]) is:

```
REQ: GET coap://[FF0X::FE]/.well-known/core?rt=mud
     Accept: TBD123456 (application/coral+cbor@identity)
```

5.1.3. Direct MUD discovery

Using [RFC6690] using CoRE Link Format, a CoAP endpoint could attempt to configure itself based on another Thing's MUD. For that reason it might fetch directly the MUD file from the device. It would start by finding if the endpoint has a MUD. The example in Link-Format [RFC6690] is:

```
REQ: GET coap://[2001:db8:3::123]:5683/.well-known/core?rt=mud

RES: 2.05 Content

     </mud/light>;rt=mud
```

In CoRAL ([I-D.ietf-core-coral], [I-D.hartke-t2trg-coral-reef]):

```
REQ: GET coap://[2001:db8:3::123]:5683/.well-known/core?rt=mud
     Accept: TBD123456 (application/coral+cbor@identity)

RES: 2.05 Content
     Content-Format: TBD123456 (application/coral+cbor@identity)

     rd-item </mud/light> { rt "mud" }
```

Once the client knows that there is a MUD file under "/mud/lightmud", it can decide to follow the presented links and query it.

```
REQ: GET coap://[2001:db8:3::123]:5683/mud/light
RES: 2.05 Content

     [{MUD Payload in SENML}]
```

In CoRAL ([I-D.ietf-core-coral], [I-D.hartke-t2trg-coral-reef]):

```
REQ: GET coap://[2001:db8:3::123]:5683/mud/light

RES: 2.05 Content

     [{MUD Payload in SENML}]
```

The device may also observe the MUD resource using [RFC7641], directly subscribing to future network configuration changes.

6. MUD File

TBD. Behaviors that are specific of CoAP should be here.

6.1. Serialization

TBD. Write about SenML/CBOR MUDs.

7. Security Considerations

TBD.

Things will expose a MUD file that has to be be signed both by the MUD author and by the device operator. Security Considerations present on Section 4.1 of [RFC8576].

We might want to use BRSKI or another similar mechanism.

Optionally the device could advertise localhost on the URL with the path to the MUD. When the network has the IP it'd append the path to it in order to fetch the MUD.

If the host part is always dynamically computed how are bootstrap / attachment schemes that depend on certs (EAP) work?

8. IANA Considerations

TBD: rt=mud should be registered.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <https://www.rfc-editor.org/info/rfc6775>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

9.2. Informative References

[I-D.hartke-t2trg-coral-reef]
Hartke, K., "Resource Discovery in Constrained RESTful Environments (CoRE) using the Constrained RESTful Application Language (CoRAL)", draft-hartke-t2trg-coral-reef-03 (work in progress), November 2019.

[I-D.ietf-core-coral]
Hartke, K., "The Constrained RESTful Application Language (CoRAL)", draft-ietf-core-coral-02 (work in progress), January 2020.

[I-D.ietf-core-resource-directory]
Shelby, Z., Koster, M., Bormann, C., Stok, P., and C. Amsuess, "CoRE Resource Directory", draft-ietf-core-resource-directory-23 (work in progress), July 2019.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/info/rfc6690>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, DOI 10.17487/RFC8576, April 2019, <https://www.rfc-editor.org/info/rfc8576>.

Acknowledgments

Thanks to Thomas Fossati, Klaus Hartke, Carsten Bormann and Michael Richardson for discussions on the problem space and reviews.

Author's Address

Jaime Jimenez
Ericsson

Phone: +358-442-992-827
Email: jaime@iki.fi