Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 11, 2018

E. Lear
Cisco Systems
R. Droms
Google
D. Romascanu
April 09, 2018

Manufacturer Usage Description Specification
draft-ietf-opsawg-mud-20

Abstract

This memo specifies a component-based architecture for manufacturer usage descriptions (MUD). The goal of MUD is to provide a means for Things to signal to the network what sort of access and network functionality they require to properly function. The initial focus is on access control. Later work can delve into other aspects.

This memo specifies two YANG modules, IPv4 and IPv6 DHCP options, an LLDP TLV, a URL, an X.509 certificate extension and a means to sign and verify the descriptions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 11, 2018.

Table of Contents

1. Introduction

The Internet has largely been constructed for general purpose computers, those devices that may be used for a purpose that is specified by those who own the device. [RFC1984] presumed that an end device would be most capable of protecting itself. This made sense when the typical device was a workstation or a mainframe, and it continues to make sense for general purpose computing devices today, including laptops, smart phones, and tablets.

[RFC7452] discusses design patterns for, and poses questions about, smart objects. Let us then posit a group of objects that are specifically not general purpose computers. These devices, which this memo refers to as Things, have a specific purpose. By definition, therefore, all other uses are not intended. The combination of these two statements can be restated as a manufacturer usage description (MUD) that can be applied at various points within a network.

We use the notion of "manufacturer" loosely in this context to refer to the entity or organization that will state how a device is intended to be used. For example, in the context of a lightbulb,

this might indeed be the lightbulb manufacturer. In the context of a smarter device that has a built in Linux stack, it might be an integrator of that device. The key points are that the device itself is assumed to serve a limited purpose, and that there may exist an organization in the supply chain of that device that will take responsibility for informing the network about that purpose.

The intent of MUD is to provide the following:

- Substantially reduce the threat surface on a device entering a network to those communications intended by the manufacturer.
- Provide a means to scale network policies to the ever-increasing number of types of devices in the network.
- Provide a means to address at least some vulnerabilities in a way that is faster than the time it might take to update systems. This will be particularly true for systems that are no longer supported by their manufacturer.
- Keep the cost of implementation of such a system to the bare minimum.
- Provide a means of extensibility for manufacturers to express other device capabilities or requirements.

MUD consists of three architectural building blocks:

- A URL that is can be used to locate a description;
- The description itself, including how it is interpreted, and;
- A means for local network management systems to retrieve the description.

In this specification we describe each of these building blocks and how they are intended to be used together. However, they may also be used separately, independent of this specification, by local deployments for their own purposes.

1.1. What MUD Doesn't Do

MUD is not intended to address network authorization of general purpose computers, as their manufacturers cannot envision a specific communication pattern to describe. In addition, even those devices that have a single or small number of uses might have very broad communication patterns. MUD on its own is not for them either.

Although MUD can provide network administrators with some additional protection when device vulnerabilities exist, it will never replace the need for manufacturers to patch vulnerabilities.

Finally, no matter what the manufacturer specifies in a MUD file, these are not directives, but suggestions. How they are instantiated locally will depend on many factors and will be ultimately up to the local network administrator, who must decide what is appropriate in a given circumstances.

1.2. A Simple Example

A light bulb is intended to light a room. It may be remotely controlled through the network, and it may make use of a rendezvous service of some form that an application on a smart phone. What we can say about that light bulb, then, is that all other network access is unwanted. It will not contact a news service, nor speak to the refrigerator, and it has no need of a printer or other devices. It has no social networking friends. Therefore, an access list applied to it that states that it will only connect to the single rendezvous service will not impede the light bulb in performing its function, while at the same time allowing the network to provide both it and other devices an additional layer of protection.

1.3. Terminology

MUD: manufacturer usage description.

MUD file: a file containing YANG-based JSON that describes a Thing and associated suggested specific network behavior.

MUD file server: a web server that hosts a MUD file.

MUD controller: the system that requests and receives the MUD file from the MUD server. After it has processed a MUD file, it may direct changes to relevant network elements.

MUD URL: a URL that can be used by the MUD controller to receive the MUD file.

Thing: the device emitting a MUD URL.

Manufacturer: the entity that configures the Thing to emit the MUD URL and the one who asserts a recommendation in a MUD file. The manufacturer might not always be the entity that constructs a Thing. It could, for instance, be a systems integrator, or even a component provider.

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].

1.4. Determining Intended Use

The notion of intended use is in itself not new. Network administrators apply access lists every day to allow for only such use. This notion of white listing was well described by Chapman and Zwicky in [FW95]. Profiling systems that make use of heuristics to identify types of systems have existed for years as well.

A Thing could just as easily tell the network what sort of access it requires without going into what sort of system it is. This would, in effect, be the converse of [RFC7488]. In seeking a general purpose solution, however, we assume that a device has so few capabilities that it will implement the least necessary capabilities to function properly. This is a basic economic constraint. Unless the network would refuse access to such a device, its developers would have no reason to provide the network any information. To date, such an assertion has held true.

1.5. Finding A Policy: The MUD URL

Our work begins with the device emitting a Universal Resource Locator (URL) [RFC3986]. This URL serves both to classify the device type and to provide a means to locate a policy file.

MUD URLs MUST use the HTTPS scheme [RFC7230].

In this memo three means are defined to emit the MUD URL, as follows:

- A DHCP option[RFC2131],[RFC3315] that the DHCP client uses to inform the DHCP server. The DHCP server may take further actions, such as retrieve the URL or otherwise pass it along to network management system or controller.

- An X.509 constraint. The IEEE has developed [IEEE8021AR] that provides a certificate-based approach to communicate device characteristics, which itself relies on [RFC5280]. The MUD URL extension is non-critical, as required by IEEE 802.1AR. Various means may be used to communicate that certificate, including Tunnel Extensible Authentication Protocol (TEAP) [RFC7170].

- Finally, a Link Layer Discovery Protocol (LLDP) frame is defined [IEEE8021AB].

It is possible that there may be other means for a MUD URL to be learned by a network. For instance, some devices may already be fielded or have very limited ability to communicate a MUD URL, and yet can be identified through some means, such as a serial number or a public key. In these cases, manufacturers may be able to map those identifiers to particular MUD URLs (or even the files themselves). Similarly, there may be alternative resolution mechanisms available for situations where Internet connectivity is limited or does not exist. Such mechanisms are not described in this memo, but are possible. Implementors should allow for this sort of flexibility of how MUD URLs may be learned.

1.6. Processing of the MUD URL

MUD controllers that are able to do so SHOULD retrieve MUD URLs and signature files as per [RFC7230], using the GET method [RFC7231]. They MUST validate the certificate using the rules in [RFC2618], Section 3.1.

Requests for MUD URLs SHOULD include an "Accept" header ([RFC7231], Section 5.3.2) containing "application/mud+json", an "Accept-Language" header ([RFC7231], Section 5.3.5), and a "User-Agent" header ([RFC7231], Section 5.5.3).

MUD controllers SHOULD automatically process 3xx response status codes.

If a MUD controller is not able to fetch a MUD URL, other means MAY be used to import MUD files and associated signature files. So long as the signature of the file can be validated, the file can be used. In such environments, controllers SHOULD warn administrators when cache-validity expiry is approaching so that they may check for new files.

1.7. Types of Policies

When the MUD URL is resolved, the MUD controller retrieves a file that describes what sort of communications a device is designed to have. The manufacturer may specify either specific hosts for cloud based services or certain classes for access within an operational network. An example of a class might be "devices of a specified manufacturer type", where the manufacturer type itself is indicated simply by the authority component (e.g, the domain name) of the MUD URL. Another example might be to allow or disallow local access. Just like other policies, these may be combined. For example:

- Allow access to devices of the same manufacturer

- Allow access to and from controllers via Constrained Application Protocol (COAP)[RFC7252]

- Allow access to local DNS/NTP

- Deny all other access

A printer might have a description that states:

- Allow access for port IPP or port LPD

- Allow local access for port HTTP

- Deny all other access

In this way anyone can print to the printer, but local access would be required for the management interface.

The files that are retrieved are intended to be closely aligned to existing network architectures so that they are easy to deploy. We make use of YANG [RFC7950] because of the time and effort spent to develop accurate and adequate models for use by network devices. JSON is used as a serialization for compactness and readability, relative to XML. Other formats may be chosen with later versions of MUD.

While the policy examples given here focus on access control, this is not intended to be the sole focus. By structuring the model described in this document with clear extension points, other descriptions could be included. One that often comes to mind is quality of service.

The YANG modules specified here are extensions of [I-D.ietf-netmod-acl-model]. The extensions to this model allow for a manufacturer to express classes of systems that a manufacturer would find necessary for the proper function of the device. Two modules are specified. The first module specifies a means for domain names to be used in ACLs so that devices that have their controllers in the cloud may be appropriately authorized with domain names, where the mapping of those names to addresses may rapidly change.

The other module abstracts away IP addresses into certain classes that are instantiated into actual IP addresses through local processing. Through these classes, manufacturers can specify how the device is designed to communicate, so that network elements can be configured by local systems that have local topological knowledge. That is, the deployment populates the classes that the manufacturer

specifies. The abstractions below map to zero or more hosts, as follows:

Manufacturer: A device made by a particular manufacturer, as identified by the authority component of its MUD URL

same-manufacturer: Devices that have the same authority component of their MUD URL.

controller: Devices that the local network administrator admits to the particular class.

my-controller: Devices associated with the MUD URL of a device that the administrator admits.

local: The class of IP addresses that are scoped within some administrative boundary. By default it is suggested that this be the local subnet.

The "manufacturer" classes can be easily specified by the manufacturer, whereas controller classes are initially envisioned to be specified by the administrator.

Because manufacturers do not know who will be using their devices, it is important for functionality referenced in usage descriptions to be relatively ubiquitous and mature. For these reasons only a limited subset YANG-based configuration is permitted in a MUD file.

1.8. The Manufacturer Usage Description Architecture

With these components laid out we now have the basis for an architecture. This leads us to ASCII art.

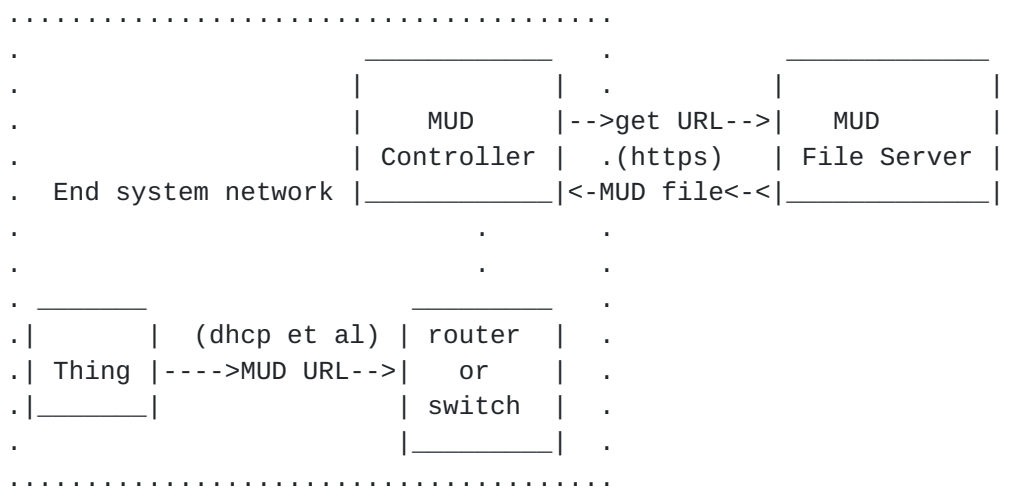


Figure 1: MUD Architecture

In the above diagram, the switch or router collects MUD URLs and forwards them to the MUD controller (a network management system) for processing. This happens in different ways, depending on how the URL is communicated. For instance, in the case of DHCP, the DHCP server might receive the URL and then process it. In the case of IEEE 802.1X, the switch would carry the URL via a certificate to the authentication server via EAP over Radius[RFC3748], which would then process it. One method to do this is TEAP, described in [RFC7170]. The certificate extension is described below.

The information returned by the MUD file server (a web server) is valid for the duration of the Thing's connection, or as specified in the description. Thus if the Thing is disconnected, any associated configuration in the switch can be removed. Similarly, from time to time the description may be refreshed, based on new capabilities or communication patterns or vulnerabilities.

The web server is typically run by or on behalf of the manufacturer. Its domain name is that of the authority found in the MUD URL. For legacy cases where Things cannot emit a URL, if the switch is able to determine the appropriate URL, it may proxy it, the trivial cases being a hardcoded MUD-URL on a switch port, or a mapping from some available identifier such as an L2 address or certificate hash to a MUD-URL.

The role of the MUD controller in this environment is to do the following:

- receive MUD URLs,

- fetch MUD files,

- translate abstractions in the MUD files to specific network element configuration,
- maintain and update any required mappings of the abstractions, and
- update network elements with appropriate configuration.

A MUD controller may be a component of a AAA or network management system. Communication within those systems and from those systems to network elements is beyond the scope of this memo.

1.9. Order of operations

As mentioned above, MUD contains architectural building blocks, and so order of operation may vary. However, here is one clear intended example:

1. Thing emits URL.
2. That URL is forwarded to a MUD controller by the nearest switch (how this happens depends on the way in which the MUD URL is emitted).
3. The MUD controller retrieves the MUD file and signature from the MUD file server, assuming it doesn't already have copies. After validating the signature, it may test the URL against a web or domain reputation service, and it may test any hosts within the file against those reputation services, as it deems fit.
4. The MUD controller may query the administrator for permission to add the Thing and associated policy. If the Thing is known or the Thing type is known, it may skip this step.
5. The MUD controller instantiates local configuration based on the abstractions defined in this document.
6. The MUD controller configures the switch nearest the Thing. Other systems may be configured as well.
7. When the Thing disconnects, policy is removed.

2. The MUD Model and Semantic Meaning

A MUD file consists of a YANG model that has been serialized in JSON [RFC7951]. For purposes of MUD, the nodes that can be modified are access lists as augmented by this model. The MUD file is limited to the serialization of only the following YANG schema:

- ietf-access-control-list [I-D.ietf-netmod-acl-model]

- ietf-mud (this document)

- ietf-acldns (this document)

Extensions may be used to add additional schema. This is described further on.

To provide the widest possible deployment, publishers of MUD files SHOULD make use of the abstractions in this memo and avoid the use of IP addresses. A MUD controller SHOULD NOT automatically implement any MUD file that contains IP addresses, especially those that might have local significance. The addressing of one side of an access list is implicit, based on whether it is applied as to-device-policy or from-device-policy.

With the exceptions of "name" of the ACL, "type", "name" of the ACE, and TCP and UDP source and destination port information, publishers of MUD files SHOULD limit the use of ACL model leaf nodes expressed to those found in this specification. Absent any extensions, MUD files are assumed to implement only the following ACL model features:

- match-on-ipv4, match-on-ipv6, match-on-tcp, match-on-udp, match-on-icmp

Furthermore, only "accept" or "drop" actions SHOULD be included. A MUD controller MAY choose to interpret "reject" as "drop". A MUD controller SHOULD ignore all other actions. This is because manufacturers do not have sufficient context within a local deployment to know whether reject is appropriate. That is a decision that should be left to a network administrator.

Given that MUD does not deal with interfaces, the support of the "ietf-interfaces" module [RFC7223] is not required. Specifically, the support of interface-related features and branches (e.g., interface-attachment and interface-stats) of the ACL YANG module is not required.

In fact, MUD controllers MAY ignore any particular component of a description or MAY ignore the description in its entirety, and SHOULD carefully inspect all MUD descriptions. Publishers of MUD files MUST NOT include other nodes except as described in Section 3.8. See that section for more information.

2.1. The IETF-MUD YANG Module

This module is structured into three parts:

- The first container "mud" holds information that is relevant to retrieval and validity of the MUD file itself, as well as policy intended to and from the Thing.

- The second component augments the matching container of the ACL model to add several nodes that are relevant to the MUD URL, or otherwise abstracted for use within a local environment.

- The third component augments the tcp-acl container of the ACL model to add the ability to match on the direction of initiation of a TCP connection.

A valid MUD file will contain two root objects, a "mud" container and an "acls" container. Extensions may add additional root objects as required. As a reminder, when parsing access-lists, elements within a "match" block are logically ANDed. In general, a single abstraction in a match statement should be used. For instance, it makes little sense to match both "my-controller" and "controller" with an argument, since they are highly unlikely to be the same value.

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is explained in [I-D.ietf-netmod-yang-tree-diagrams].

```
module: ietf-mud
    +--rw mud!
       +--rw mud-version           uint8
       +--rw mud-url               inet:uri
       +--rw last-update           yang:date-and-time
       +--rw mud-signature?        inet:uri
       +--rw cache-validity?       uint8
       +--rw is-supported          boolean
       +--rw systeminfo?           string
       +--rw mfg-name?             string
       +--rw model-name?           string
       +--rw firmware-rev?         string
       +--rw software-rev?         string
       +--rw extensions*           string
       +--rw from-device-policy
       |  +--rw access-lists
       |     +--rw access-list* [name]
       |        +--rw name    -> /acl:access-lists/acl/name
       +--rw to-device-policy
          +--rw access-lists
             +--rw access-list* [name]
                +--rw name    -> /acl:access-lists/acl/name
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw mud
       +--rw manufacturer?        inet:host
       +--rw same-manufacturer?   empty
       +--rw model?               inet:uri
       +--rw local-networks?      empty
       +--rw controller?          inet:uri
       +--rw my-controller?       empty
  augment /acl:access-lists/acl:acl/acl:aces/acl:ace
     /acl:matches/acl:l4/acl:tcp/acl:tcp:
    +--rw direction-initiated?   direction
```

3. Data Node Definitions

Note that in this section, when we use the term "match" we are referring to the ACL model "matches" node.

The following nodes are defined.

3.1. mud-version

This node specifies the integer version of the MUD specification. This memo specifies version 1.

3.2. to-device-policy and from-device-policy containers

[I-D.ietf-netmod-acl-model] describes access-lists. In the case of MUD, a MUD file must be explicit in describing the communication pattern of a Thing, and that includes indicating what is to be permitted or denied in either direction of communication. Hence each of these containers indicates the appropriate direction of a flow in association with a particular Thing. They contain references to specific access-lists.

3.3. last-update

This is a date-and-time value of when the MUD file was generated. This is akin to a version number. Its form is taken from [RFC6991] which, for those keeping score, in turn was taken from Section 5.6 of [RFC3339], which was taken from [ISO.8601.1988].

3.4. cache-validity

This uint8 is the period of time in hours that a network management station MUST wait since its last retrieval before checking for an update. It is RECOMMENDED that this value be no less than 24 and MUST NOT be more than 168 for any Thing that is supported. This period SHOULD be no shorter than any period determined through HTTP caching directives (e.g., "cache-control" or "Expires"). N.B., expiring of this timer does not require the MUD controller to discard the MUD file, nor terminate access to a Thing. See Section 15 for more information.

3.5. is-supported

This boolean is an indication from the manufacturer to the network administrator as to whether or not the Thing is supported. In this context a Thing is said to not be supported if the manufacturer intends never to issue an update to the Thing or never update the MUD file. A MUD controller MAY still periodically check for updates.

3.6. systeminfo

This is a textual UTF-8 description of the Thing to be connected. The intent is for administrators to be able to see a localized name associated with the Thing. It SHOULD NOT exceed 60 characters worth of display space (that is- what the administrator actually sees).

3.7. mfg-name, software-rev, model-name firmware-rev

These optional fields are filled in as specified by [I-D.ietf-netmod-entity]. Note that firmware-rev and software-rev MUST NOT be populated in a MUD file if the device can be upgraded but the MUD-URL cannot be. This would be the case, for instance, with MUR-URLs that are contained in 802.1AR certificates.

3.8. extensions

This optional leaf-list names MUD extensions that are used in the MUD file. Note that NO MUD extensions may be used in a MUD file without the extensions being declared. Implementations MUST ignore any node in this file that they do not understand.

Note that extensions can either extend the MUD file as described in the previous paragraph, or they might reference other work. An extension example can be found in Appendix C.

3.9. manufacturer

This node consists of a hostname that would be matched against the authority component of another Thing's MUD URL. In its simplest form "manufacturer" and "same-manufacturer" may be implemented as access-lists. In more complex forms, additional network capabilities may be used. For example, if one saw the line "manufacturer" : "flobbidy.example.com", then all Things that registered with a MUD URL that contained flobbity.example.com in its authority section would match.

3.10. same-manufacturer

This null-valued node is an equivalent for when the manufacturer element is used to indicate the authority that is found in another Thing's MUD URL matches that of the authority found in this Thing's MUD URL. For example, if the Thing's MUD URL were https://b1.example.com/ThingV1, then all devices that had MUD URL with an authority section of b1.example.com would match.

3.11. model

This string matches the entire MUD URL, thus covering the model that is unique within the context of the authority. It may contain not only model information, but versioning information as well, and any other information that the manufacturer wishes to add. The intended use is for devices of this precise class to match, to permit or deny communication between one another.

3.12. local-networks

This null-valued node expands to include local networks. Its default expansion is that packets must not traverse toward a default route that is received from the router. However, administrators may expand the expression as is appropriate in their deployments.

3.13. controller

This URI specifies a value that a controller will register with the MUD controller. The node then is expanded to the set of hosts that are so registered. This node may also be a URN. In this case, the URN describes a well known service, such as DNS or NTP.

Great care should be used when invoking the controller class. For one thing, it requires some understanding by the administrator as to when it is appropriate. Classes that are standardized may make it possible to easily name devices that support standard functions. For instance, the MUD controller could have some knowledge of which DNS servers should be used for any particular group of Things. Non-standard classes will likely require some sort of administrator interaction. Pre-registration in such classes by controllers with the MUD server is encouraged. The mechanism to do that is beyond the scope of this work.

Controller URIs MAY take the form of a URL (e.g. "http[s]://"). However, MUD controllers MUST NOT resolve and retrieve such files, and it is RECOMMENDED that there be no such file at this time, as their form and function may be defined at a point in the future. For now, URLs should serve simply as class names and may be populated by the local deployment administrator.

3.14. my-controller

This null-valued node signals to the MUD controller to use whatever mapping it has for this MUD URL to a particular group of hosts. This may require prompting the administrator for class members. Future work should seek to automate membership management.

3.15. direction-initiated

When applied this matches packets when the flow was initiated in the corresponding direction. [RFC6092] specifies IPv6 guidance best practices. While that document is scoped specifically to IPv6, its contents are applicable for IPv4 as well. When this flag is set, and the system has no reason to believe a flow has been initiated it MUST drop the packet. This node may be implemented in its simplest form

by looking at naked SYN bits, but may also be implemented through more stateful mechanisms.

4. Processing of the MUD file

To keep things relatively simple in addition to whatever definitions exist, we also apply two additional default behaviors:

- Anything not explicitly permitted is denied.
- Local DNS and NTP are, by default, permitted to and from the Thing.

An explicit description of the defaults can be found in Appendix B. These are applied AFTER all other explicit rules. Thus, a default behavior can be changed with a "drop" action.

5. What does a MUD URL look like?

MUD URLs are required to use the HTTPS scheme, in order to establish the MUD file server's identity and assure integrity of the MUD file.

Any "https://" URL can be a MUD URL. For example:

```
https://things.example.org/product_abc123/v5
https://www.example.net/mudfiles/temperature_sensor/
https://example.com/lightbulbs/colour/v1
```

The MUD URL identifies a Thing with a specificity according to the manufacturer's wishes. It could include a brand name, model number, or something more specific. It also could provide a means to indicate what version the product is.

Specifically, if the intended communication patterns of a Thing change, as compared to other things, the MUD URL should change. For example, if a new model of light bulb is released that requires access to different network services, it would have a separate MUD URL from those that do not.

6. The MUD YANG Model

```
<CODE BEGINS>file "ietf-mud@2018-03-01.yang"
module ietf-mud {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud";
  prefix ietf-mud;

  import ietf-access-control-list {
```

```
      prefix acl;
    }
    import ietf-yang-types {
      prefix yang;
    }
    import ietf-inet-types {
      prefix inet;
    }

    organization
      "IETF OPSAWG (Ops Area) Working Group";
    contact
      "WG Web: http://tools.ietf.org/wg/opsawg/
       WG List: opsawg@ietf.org
       Author: Eliot Lear
       lear@cisco.com
       Author: Ralph Droms
       rdroms@gmail.com
       Author: Dan Romascanu
       dromasca@gmail.com

      ";
    description
      "This YANG module defines a component that augments the
       IETF description of an access list.  This specific module
       focuses on additional filters that include local, model,
       and same-manufacturer.

       This module is intended to be serialized via JSON and stored
       as a file, as described in RFC XXXX [RFC Editor to fill in with
       this document #].

       Copyright (c) 2016,2017 IETF Trust and the persons
       identified as the document authors.  All rights reserved.
       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD
       License set forth in Section 4.c of the IETF Trust's Legal
       Provisions Relating to IETF Documents
       (http://trustee.ietf.org/license-info).
       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    revision 2018-03-01 {
      description
        "Initial proposed standard.";
      reference
        "RFC XXXX: Manufacturer Usage Description
```

```
          Specification";
      }

      typedef direction {
        type enumeration {
          enum "to-device" {
            description
              "packets or flows destined to the target
               Thing";
          }
          enum "from-device" {
            description
              "packets or flows destined from
               the target Thing";
          }
        }
        description
          "Which way are we talking about?";
      }

      container mud {
        presence "Enabled for this particular MUD URL";
        description
          "MUD related information, as specified
           by RFC-XXXX [RFC Editor to fill in].";
        uses mud-grouping;
      }

      grouping mud-grouping {
        description
          "Information about when support end(ed), and
           when to refresh";

        leaf mud-version {
          type uint8;
          mandatory true;
          description "This is the version of the MUD
             specification.  This memo specifies version 1.";
         }


        leaf mud-url {
          type inet:uri;
          mandatory true;
          description
            "This is the MUD URL associated with the entry found
             in a MUD file.";
        }
```

```
       leaf last-update {
         type yang:date-and-time;
         mandatory true;
         description
           "This is intended to be when the current MUD file
            was generated.  MUD Controllers SHOULD NOT check
            for updates between this time plus cache validity";
       }

       leaf mud-signature {
         type inet:uri;
         description "A URI that resolves to a signature as
           described in this specification.";
       }

       leaf cache-validity {
         type uint8 {
           range "1..168";
         }
         units "hours";
         default "48";
         description
           "The information retrieved from the MUD server is
            valid for these many hours, after which it should
            be refreshed.  N.B. MUD controller implementations
            need not discard MUD files beyond this period.";
       }
       leaf is-supported {
         type boolean;
         mandatory true;
         description
           "This boolean indicates whether or not the Thing is
            currently supported by the manufacturer.";
       }
       leaf systeminfo {
         type string;
         description
           "A UTF-8 description of this Thing.  This
            should be a brief description that may be
            displayed to the user to determine whether
            to allow the Thing on the
            network.";
       }

       leaf mfg-name {
            type string;
            description "Manufacturer name, as described in
               the ietf-hardware yang module.";
```

```
      }

      leaf model-name {
           type string;
           description "Model name, as described in the
           ietf-hardware yang module.";
      }

      leaf firmware-rev {
           type string;
           description "firmware-rev, as described in the
           ietf-hardware yang module.  Note this field MUST
           NOT be included when the device can be updated
           but the MUD-URL cannot.";
      }

      leaf software-rev {
           type string;
           description "software-rev, as described in the
           ietf-hardware yang module.  Note this field MUST
           NOT be included when the device can be updated
           but the MUD-URL cannot.";
      }

      leaf-list extensions {
        type string {
           length "1..40";
        }
        description
          "A list of extension names that are used in this MUD
           file.  Each name is registered with the IANA and
           described in an RFC.";
      }
      container from-device-policy {
        description
          "The policies that should be enforced on traffic
           coming from the device. These policies are not
           necessarily intended to be enforced at a single
           point, but may be rendered by the controller to any
           relevant enorcement points in the network or
           elsewhere.";
        uses access-lists;
      }
      container to-device-policy {
        description
          "The policies that should be enforced on traffic
           going to the device. These policies are not
           necessarily intended to be enforced at a single
```

```
          point, but may be rendered by the controller to any
          relevant enorcement points in the network or
          elsewhere.";
        uses access-lists;
      }
    }

    grouping access-lists {
      description
        "A grouping for access lists in the context of device
         policy.";
      container access-lists {
        description
          "The access lists that should be applied to traffic
             to or from the device.";
        list access-list {
          key "name";
          description
            "Each entry on this list refers to an ACL that
               should be present in the overall access list
               data model. Each ACL is identified by name and
               type.";
          leaf name {
            type leafref {
              path "/acl:acls/acl:acl/acl:name";
            }
            description
              "The name of the ACL for this entry.";
          }
        }
      }
    }
    augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {
      description
        "adding abstractions to avoid need of IP addresses";
      container mud {
        description
          "MUD-specific matches.";
        leaf manufacturer {
          type inet:host;
          description
            "A domain that is intended to match the authority
             section of the MUD URL. This node is used to specify
             one or more manufacturers a device should
             be authorized to access.";
        }
        leaf same-manufacturer {
          type empty;
```

```
          description
            "This node matches the authority section of the MUD URL
             of a Thing.  It is intended to grant access to all
             devices with the same authority section.";
        }
        leaf model {
          type inet:uri;
          description
            "Devices of the specified  model type will match if
             they have an identical MUD URL.";
        }
        leaf local-networks {
          type empty;
          description
            "IP addresses will match this node if they are
             considered local addresses.  A local address may be
             a list of locally defined prefixes and masks
             that indicate a particular administrative scope.";
        }
        leaf controller {
          type inet:uri;
          description
            "This node names a class that has associated with it
             zero or more IP addresses to match against.  These
             may be scoped to a manufacturer or via a standard
             URN.";
        }
        leaf my-controller {
          type empty;
          description
            "This node matches one or more network elements that
             have been configured to be the controller for this
             Thing, based on its MUD URL.";
        }
      }
    }
    augment "/acl:acls/acl:acl/acl:aces/" +
       "acl:ace/acl:matches/acl:l4/acl:tcp/acl:tcp" {
      description
        "add direction-initiated";
      leaf direction-initiated {
        type direction;
        description
          "This node matches based on which direction a
           connection was initiated. The means by which that
           is determined is discussed in this document.";
      }
    }
```

```
}

<CODE ENDS>
```

7. The Domain Name Extension to the ACL Model

This module specifies an extension to IETF-ACL model such that domain names may be referenced by augmenting the "matches" node. Different implementations may deploy differing methods to maintain the mapping between IP address and domain name, if indeed any are needed. However, the intent is that resources that are referred to using a name should be authorized (or not) within an access list.

The structure of the change is as follows:

```
module: ietf-acldns
  augment /acl:access-lists/acl:acl/acl:aces/acl:ace/
    acl:matches/acl:l3/acl:ipv4/acl:ipv4:
    +--rw src-dnsname?   inet:host
    +--rw dst-dnsname?   inet:host
  augment /acl:access-lists/acl:acl/acl:aces/acl:ace/
    acl:matches/acl:l3/acl:ipv6/acl:ipv6:
    +--rw src-dnsname?   inet:host
    +--rw dst-dnsname?   inet:host
```

The choice of these particular points in the access-list model is based on the assumption that we are in some way referring to IP-related resources, as that is what the DNS returns. A domain name in our context is defined in [RFC6991]. The augmentations are replicated across IPv4 and IPv6 to allow MUD file authors the ability to control the IP version that the Thing may utilize.

The following node are defined.

7.1. src-dnsname

The argument corresponds to a domain name of a source as specified by inet:host. A number of means may be used to resolve hosts. What is important is that such resolutions be consistent with ACLs required by Things to properly operate.

7.2. dst-dnsname

The argument corresponds to a domain name of a destination as specified by inet:host See the previous section relating to resolution.

Note when using either of these with a MUD file, because access is associated with a particular Thing, MUD files MUST not contain either a src-dnsname in an ACL associated with from-device-policy or a dst-dnsname associated with to-device-policy.

7.3. The ietf-acldns Model

```
<CODE BEGINS>file "ietf-acldns@2018-03-01.yang"
module ietf-acldns {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-acldns";
  prefix "ietf-acldns";

  import ietf-access-control-list {
    prefix "acl";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";

  contact
       "WG Web: http://tools.ietf.org/wg/opsawg/
       WG List: opsawg@ietf.org
       Author: Eliot Lear
       lear@cisco.com
       Author: Ralph Droms
       rdroms@gmail.com
       Author: Dan Romascanu
       dromasca@gmail.com
    ";

  description
    "This YANG module defines a component that augments the
     IETF description of an access list to allow dns names
     as matching criteria.";

  revision 2018-03-01 {
    description "Base version of dnsname extension of ACL model";
    reference "RFC XXXX: Manufacturer Usage Description
               Specification";
  }

  grouping dns-matches {
    description "Domain names for matching.";
```

```
      leaf src-dnsname {
        type inet:host;
        description "domain name to be matched against";
      }
      leaf dst-dnsname {
        type inet:host;
        description "domain name to be matched against";
      }
    }

    augment "/acl:acls/acl:acl/acl:aces/acl:ace/" +
       "acl:matches/acl:l3/acl:ipv4/acl:ipv4" {
      description "Adding domain names to matching";
      uses dns-matches;
    }

    augment "/acl:acls/acl:acl/" +
       "acl:aces/acl:ace/" +
       "acl:matches/acl:l3/acl:ipv6/acl:ipv6" {
      description "Adding domain names to matching";
      uses dns-matches;
    }
  }
  <CODE ENDS>
```

8. MUD File Example

This example contains two access lists that are intended to provide outbound access to a cloud service on TCP port 443.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://lighting.example.com/lightbulb2000",
    "last-update": "2018-03-02T11:20:51+01:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "The BMS Example Lightbulb",
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6fr"
          }
        ]
      }
    },
```

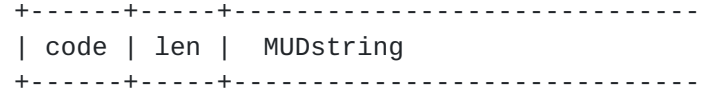
```
      "to-device-policy": {
        "access-lists": {
          "access-list": [
            {
              "name": "mud-76100-v6to"
            }
          ]
        }
      }
    },
    "ietf-access-control-list:access-lists": {
      "acl": [
        {
          "name": "mud-76100-v6to",
          "type": "ipv6-acl-type",
          "aces": {
            "ace": [
              {
                "name": "cl0-todev",
                "matches": {
                  "ipv6": {
                    "ietf-acldns:src-dnsname": "test.com",
                    "protocol": 6
                  },
                  "tcp": {
                    "ietf-mud:direction-initiated": "from-device",
                    "source-port": {
                      "operator": "eq",
                      "port": 443
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        },
        {
          "name": "mud-76100-v6fr",
          "type": "ipv6-acl-type",
          "aces": {
            "ace": [
              {
                "name": "cl0-frdev",
                "matches": {
                  "ipv6": {
```

```
                    "ietf-acldns:dst-dnsname": "test.com",
                    "protocol": 6
                  },
                  "tcp": {
                    "ietf-mud:direction-initiated": "from-device",
                    "destination-port": {
                      "operator": "eq",
                      "port": 443
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        }
      ]
    }
  }
```

In this example, two policies are declared, one from the Thing and the other to the Thing. Each policy names an access list that applies to the Thing, and one that applies from. Within each access list, access is permitted to packets flowing to or from the Thing that can be mapped to the domain name of "service.bms.example.com". For each access list, the enforcement point should expect that the Thing initiated the connection.

9. The MUD URL DHCP Option

The IPv4 MUD URL client option has the following format:

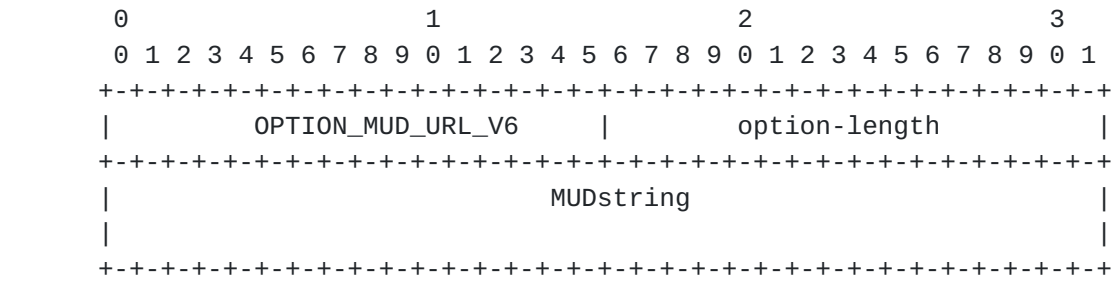


Code OPTION_MUD_URL_V4 (161) is assigned by IANA. len is a single octet that indicates the length of MUD string in octets. The MUD string is defined as follows:

```
MUDstring = mudurl [ " " reserved ]
mudurl = URI; a URL [RFC3986] that uses the "https" schema [RFC7230]
reserved = 1*( CHAR ) ; from [RFC5234]
```

The entire option MUST NOT exceed 255 octets. If a space follows the MUD URL, a reserved string that will be defined in future specifications follows. MUD controllers that do not understand this field MUST ignore it.

The IPv6 MUD URL client option has the following format:



OPTION_MUD_URL_V6 (112; assigned by IANA).

option-length contains the length of the MUDstring, as defined above, in octets.

The intent of this option is to provide both a new Thing classifier to the network as well as some recommended configuration to the routers that implement policy. However, it is entirely the purview of the network system as managed by the network administrator to decide what to do with this information. The key function of this option is simply to identify the type of Thing to the network in a structured way such that the policy can be easily found with existing toolsets.

9.1. Client Behavior

A DHCPv4 client MAY emit a DHCPv4 option and a DHCPv6 client MAY emit DHCPv6 option. These options are singletons, as specified in [RFC7227]. Because clients are intended to have at most one MUD URL associated with them, they may emit at most one MUD URL option via DHCPv4 and one MUD URL option via DHCPv6. In the case where both v4 and v6 DHCP options are emitted, the same URL MUST be used.

Clients SHOULD log or otherwise report improper acknowledgments from servers, but they MUST NOT modify their MUD URL configuration based on a server's response. The server's response is only an acknowledgment that the server has processed the option, and promises no specific network behavior to the client. In particular, it may not be possible for the server to retrieve the file associated with the MUD URL, or the local network administration may not wish to use

the usage description. Neither of these situations should be considered in any way exceptional.

9.2. Server Behavior

A DHCP server may ignore these options or take action based on receipt of these options. If a server successfully parses the option and the URL, it MUST return the option with length field set to zero and a corresponding null URL field as an acknowledgment. Even in this circumstance, no specific network behavior is guaranteed. When a server consumes this option, it will either forward the URL and relevant client information (such as the gateway address or giaddr) to a network management system, or it will retrieve the usage description itself by resolving the URL.

DHCP servers may implement MUD functionality themselves or they may pass along appropriate information to a network management system or MUD controller. A DHCP server that does process the MUD URL MUST adhere to the process specified in [RFC2818] and [RFC5280] to validate the TLS certificate of the web server hosting the MUD file. Those servers will retrieve the file, process it, create and install the necessary configuration on the relevant network element. Servers SHOULD monitor the gateway for state changes on a given interface. A DHCP server that does not provide MUD functionality and has forwarded a MUD URL to a MUD controller MUST notify the MUD controller of any corresponding change to the DHCP state of the client (such as expiration or explicit release of a network address lease).

9.3. Relay Requirements

There are no additional requirements for relays.

10. The Manufacturer Usage Description (MUD) URL X.509 Extension

This section defines an X.509 non-critical certificate extension that contains a single Uniform Resource Locator (URL) that points to an on-line Manufacturer Usage Description concerning the certificate subject. URI must be represented as described in Section 7.4 of [RFC5280].

Any Internationalized Resource Identifiers (IRIs) MUST be mapped to URIs as specified in Section 3.1 of [RFC3987] before they are placed in the certificate extension.

The semantics of the URL are defined Section 5 of this document.

The choice of id-pe is based on guidance found in Section 4.2.2 of [RFC5280]:

These extensions may be used to direct applications to on-line information about the issuer or the subject.

The MUD URL is precisely that: online information about the particular subject.

The new extension is identified as follows:

<CODE BEGINS>

```
MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
             internet(1) security(5) mechanisms(5) pkix(7)
             id-mod(0) id-mod-mudURLExtn2016(88) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS ALL --

IMPORTS
  EXTENSION
  FROM PKIX-CommonTypes-2009
         { iso(1) identified-organization(3) dod(6) internet(1)
           security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-pkixCommon-02(57) }

  id-pe
  FROM PKIX1Explicit-2009
         { iso(1) identified-organization(3) dod(6) internet(1)
           security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-pkix1-explicit-02(51) } ;
  MUDCertExtensions EXTENSION ::= { ext-MUDURL, ... }
  ext-MUDURL EXTENSION ::= { SYNTAX MUDURLSyntax
  IDENTIFIED BY id-pe-mud-url }

  id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }

  MUDURLSyntax ::= IA5String

  END
```

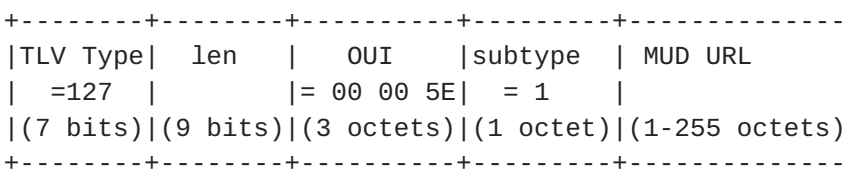
<CODE ENDS>

While this extension can appear in either an 802.AR manufacturer certificate (IDevID) or deployment certificate (LDevID), of course it is not guaranteed in either, nor is it guaranteed to be carried over. It is RECOMMENDED that MUD controller implementations maintain a table that maps a Thing to its MUD URL based on IDevIDs.

11. The Manufacturer Usage Description LLDP extension

The IEEE802.1AB Link Layer Discovery Protocol (LLDP) is a one hop vendor-neutral link layer protocol used by end hosts network Things for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network. Its Type-Length-Value (TLV) design allows for 'vendor-specific' extensions to be defined. IANA has a registered IEEE 802 organizationally unique identifier (OUI) defined as documented in [RFC7042]. The MUD LLDP extension uses a subtype defined in this document to carry the MUD URL.

The LLDP vendor specific frame has the following format:



where:

- TLV Type = 127 indicates a vendor-specific TLV
- len - indicates the TLV string length
- OUI = 00 00 5E is the organizationally unique identifier of IANA
- subtype = 1 (to be assigned by IANA for the MUD URL)
- MUD URL - the length MUST NOT exceed 255 octets

The intent of this extension is to provide both a new Thing classifier to the network as well as some recommended configuration to the routers that implement policy. However, it is entirely the purview of the network system as managed by the network administrator to decide what to do with this information. The key function of this extension is simply to identify the type of Thing to the network in a structured way such that the policy can be easily found with existing toolsets.

Hosts, routers, or other network elements that implement this option are intended to have at most one MUD URL associated with them, so they may transmit at most one MUD URL value.

Hosts, routers, or other network elements that implement this option may ignore these options or take action based on receipt of these options. For example they may fill in information in the respective

extensions of the LLDP Management Information Base (LLDP MIB). LLDP operates in a one-way direction. LLDPDUs are not exchanged as information requests by one Thing and response sent by another Thing. The other Things do not acknowledge LLDP information received from a Thing. No specific network behavior is guaranteed. When a Thing consumes this extension, it may either forward the URL and relevant remote Thing information to a MUD controller, or it will retrieve the usage description by resolving the URL in accordance with normal HTTP semantics.

12. Creating and Processing of Signed MUD Files

Because MUD files contain information that may be used to configure network access lists, they are sensitive. To insure that they have not been tampered with, it is important that they be signed. We make use of DER-encoded Cryptographic Message Syntax (CMS) [RFC5652] for this purpose.

12.1. Creating a MUD file signature

A MUD file MUST be signed using CMS as an opaque binary object. In order to make successful verification more likely, intermediate certificates SHOULD be included. The signature is stored at the location specified in the MUD file. Signatures are transferred using content-type "application/pkcs7-signature".

For example:

```
% openssl cms -sign -signer mancertfile -inkey mankey \
              -in mudfile -binary -outform DER - \
              -certfile intermediatecert -out mudfile.p7s
```

Note: A MUD file may need to be re-signed if the signature expires.

12.2. Verifying a MUD file signature

Prior to retrieving a MUD file the MUD controller SHOULD retrieve the MUD signature file by retrieving the value of "mud-signature" and validating the signature across the MUD file.

Upon retrieving a MUD file, a MUD controller MUST validate the signature of the file before continuing with further processing. A MUD controller MUST cease processing of that file it cannot validate the chain of trust to a known trust anchor until an administrator has given approval.

The purpose of the signature on the file is to assign accountability to an entity, whose reputation can be used to guide administrators on

whether or not to accept a given MUD file. It is already common place to check web reputation on the location of a server on which a file resides. While it is likely that the manufacturer will be the signer of the file, this is not strictly necessary, and may not be desirable. For one thing, in some environments, integrators may install their own certificates. For another, what is more important is the accountability of the recommendation, and not the cryptographic relationship between the device and the file.

An example:

```
% openssl cms -verify -in mudfile.p7s -inform DER -content mudfile
```

Note the additional step of verifying the common trust root.

13. Extensibility

One of our design goals is to see that MUD files are able to be understood by as broad a cross-section of systems as is possible. Coupled with the fact that we have also chosen to leverage existing mechanisms, we are left with no ability to negotiate extensions and a limited desire for those extensions in any event. A such, a two-tier extensibility framework is employed, as follows:

1. At a coarse grain, a protocol version is included in a MUD URL. This memo specifies MUD version 1. Any and all changes are entertained when this version is bumped. Transition approaches between versions would be a matter for discussion in future versions.

2. At a finer grain, only extensions that would not incur additional risk to the Thing are permitted. Specifically, adding nodes to the mud container is permitted with the understanding that such additions will be ignored by unaware implementations. Any such extensions SHALL be standardized through the IETF process, and MUST be named in the "extensions" list. MUD controllers MUST ignore YANG nodes they do not understand and SHOULD create an exception to be resolved by an administrator, so as to avoid any policy inconsistencies.

14. Deployment Considerations

Because MUD consists of a number of architectural building blocks, it is possible to assemble different deployment scenarios. One key aspect is where to place policy enforcement. In order to protect the Thing from other Things within a local deployment, policy can be enforced on the nearest switch or access point. In order to limit unwanted traffic within a network, it may also be advisable to

enforce policy as close to the Internet as possible. In some circumstances, policy enforcement may not be available at the closest hop. At that point, the risk of so-called east-west infection is increased to the number of Things that are able to communicate without protection.

A caution about some of the classes: admission of a Thing into the "manufacturer" and "same-manufacturer" class may have impact on access of other Things. Put another way, the admission may grow the access-list on switches connected to other Things, depending on how access is managed. Some care should be given on managing that access-list growth. Alternative methods such as additional network segmentation can be used to keep that growth within reason.

Because as of this writing MUD is a new concept, one can expect a great many devices to not have implemented it. It remains a local deployment decision as to whether a device that is first connected should be alloewed broad or limited access. Furthermore, as mentioned in the introduction, a deployment may choose to ignore a MUD policy in its entirety, but simply taken into account the MUD URL as a classifier to be used as part of a local policy decision.

15. Security Considerations

Based on how a MUD URL is emitted, a Thing may be able to lie about what it is, thus gaining additional network access. There are several means to limit risk in this case. The most obvious is to only believe Things that make use of certificate-based authentication such as IEEE 802.1AR certificates. When those certificates are not present, Things claiming to be of a certain manufacturer SHOULD NOT be included in that manufacturer grouping without additional validation of some form. This will occur when it makes use of primitives such as "manufacturer" for the purpose of accessing Things of a particular type. Similarly, network management systems may be able to fingerprint the Thing. In such cases, the MUD URL can act as a classifier that can be proven or disproven. Fingerprinting may have other advantages as well: when 802.1AR certificates are used, because they themselves cannot change, fingerprinting offers the opportunity to add artificats to the MUD URL. The meaning of such artifacts is left as future work.

Network management systems SHOULD NOT accept a usage description for a Thing with the same MAC address that has indicated a change of authority without some additional validation (such as review by a network administrator). New Things that present some form of unauthenticated MUD URL SHOULD be validated by some external means when they would be otherwise be given increased network access.

It may be possible for a rogue manufacturer to inappropriately exercise the MUD file parser, in order to exploit a vulnerability. There are three recommended approaches to address this threat. The first is to validate the signature of the MUD file. The second is to have a system do a primary scan of the file to ensure that it is both parseable and believable at some level. MUD files will likely be relatively small, to start with. The number of ACEs used by any given Thing should be relatively small as well. It may also be useful to limit retrieval of MUD URLs to only those sites that are known to have decent web or domain reputations.

Use of a URL necessitates the use of domain names. If a domain name changes ownership, the new owner of that domain may be able to provide MUD files that MUD controllers would consider valid. There are a few approaches that can mitigate this attack. First, MUD controllers SHOULD cache certificates used by the MUD file server. When a new certificate is retrieved for whatever reason, the MUD controller should check to see if ownership of the domain has changed. A fair programmatic approximation of this is when the name servers for the domain have changed. If the actual MUD file has changed, the controller MAY check the WHOIS database to see if registration ownership of a domain has changed. If a change has occured, or if for some reason it is not possible to determine whether ownership has changed, further review may be warranted. Note, this remediation does not take into account the case of a Thing that was produced long ago and only recently fielded, or the case where a new MUD controller has been installed.

It may not be possible for a MUD controller to retrieve a MUD file at any given time. Should a MUD controller fail to retrieve a MUD file, it SHOULD consider the existing one safe to use, at least for a time. After some period, it SHOULD log that it has been unable to retrieve the file. There may be very good reasons for such failures, including the possibility that the MUD controller is in an off-line environment, the local Internet connection has failed, or the remote Internet connection has failed. It is also possible that an attacker is attempting to prevent onboarding of a device. It is a local deployment decision as to whether or not devices may be onboarded in the face of such failures.

The release of a MUD URL by a Thing reveals what the Thing is, and provides an attacker with guidance on what vulnerabilities may be present.

While the MUD URL itself is not intended to be unique to a specific Thing, the release of the URL may aid an observer in identifying individuals when combined with other information. This is a privacy consideration.

In addressing both of these concerns, implementors should take into account what other information they are advertising through mechanisms such as mDNS[RFC6872], how a Thing might otherwise be identified, perhaps through how it behaves when it is connected to the network, whether a Thing is intended to be used by individuals or carry personal identifying information, and then apply appropriate data minimization techniques. One approach is to make use of TEAP [RFC7170] as the means to share information with authorized components in the network. Network elements may also assist in limiting access to the MUD URL through the use of mechanisms such as DHCPv6-Shield [RFC7610].

Please note that the security considerations mentioned in Section 4.7 of [I-D.ietf-netmod-rfc6087bis] are not applicable in this case because the YANG serialization is not intended to be accessed via NETCONF. However, for those who try to instantiate this model in a network element via NETCONF, all objects in each model in this draft exhibit similar security characteristics as [I-D.ietf-netmod-acl-model]. The basic purpose of MUD is to configure access, and so by its very nature can be disruptive if used by unauthorized parties.

16. IANA Considerations

16.1. YANG Module Registrations

The following YANG modules are requested to be registred in the "IANA Module Names" registry:

The ietf-mud module:

- Name: ietf-mud
- XML Namespace: urn:ietf:params:xml:ns:yang:ietf-mud
- Prefix: ief-mud
- Reference: This memo

The ietf-acldns module:

- Name: ietf-acldns
- XML Namespace: urn:ietf:params:xml:ns:yang:ietf-acldns
- Prefix: ietf-acldns
- Reference: This memo

16.2. DHCPv4 and DHCPv6 Options

The IANA has allocated option 161 in the Dynamic Host Configuration Protocol (DHCP) and Bootstrap Protocol (BOOTP) Parameters registry for the MUD DHCPv4 option, and option 112 for DHCPv6, as described in Section 9.

16.3. PKIX Extensions

IANA is kindly requested to make the following assignments for:

o The MUDURLExtnModule-2016 ASN.1 module in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0).

o id-pe-mud-url object identifier from the "SMI Security for PKIX Certificate Extension" registry (1.3.6.1.5.5.7.1).

The use of these values is specified in Section 10.

16.4. MIME Media-type Registration for MUD files

The following media-type is defined for transfer of MUD file:

- Type name: application
- Subtype name: mud+json
- Required parameters: n/a
- Optional parameters: n/a
- Encoding considerations: 8bit; application/mud+json values are represented as a JSON object; UTF-8 encoding SHOULD be employed.
- Security considerations: See Security Considerations of this document.
- Interoperability considerations: n/a
- Published specification: this document
- Applications that use this media type: MUD controllers as specified by this document.
- Fragment identifier considerations: n/a
- Additional information:

    Magic number(s): n/a
    File extension(s): n/a
    Macintosh file type code(s): n/a

- Person & email address to contact for further information: Eliot Lear <lear@cisco.com>, Ralph Droms <rdroms@cisco.com>
- Intended usage: COMMON
- Restrictions on usage: none
- Author:
    Eliot Lear <lear@cisco.com>
    Ralph Droms <rdroms@cisco.com>
- Change controller: IESG
- Provisional registration? (standards tree only): No.

16.5. LLDP IANA TLV Subtype Registry

IANA is requested to create a new registry for IANA Link Layer Discovery Protocol (LLDP) TLV subtype values. The recommended policy for this registry is Expert Review. The maximum number of entries in the registry is 256.

IANA is required to populate the initial registry with the value:

LLDP subtype value = 1 (All the other 255 values should be initially marked as 'Unassigned'.)

Description = the Manufacturer Usage Description (MUD) Uniform Resource Locator (URL)

Reference = < this document >

16.6. The MUD Well Known Universal Resource Name (URNs)

The following parameter registry is requested to be added in accordance with [RFC3553]

Registry name: "urn:ietf:params:mud" is requested.
Specification: this document
Repository: this document
Index value: Encoded identically to a TCP/UDP port service name, as specified in Section 5.1 of [RFC6335]

The following entries should be added to the "urn:ietf:params:mud" name space:

"urn:ietf:params:mud:dns" refers to the service specified by [RFC1123]. "urn:ietf:params:mud:ntp" refers to the service specified by [RFC5905].

16.7. Extensions Registry

The IANA is requested to establish a registry of extensions as follows:

Registry name: MUD extensions registry
Registry policy: Standards action
Standard reference: document
Extension name: UTF-8 encoded string, not to exceed 40 characters.

Each extension MUST follow the rules specified in this specification. As is usual, the IANA issues early allocations based in accordance with [RFC7120].

17. Acknowledgments

The authors would like to thank Einar Nilsen-Nygaard, who singlehandedly updated the model to match the updated ACL model, Bernie Volz, Tom Gindin, Brian Weis, Sandeep Kumar, Thorsten Dahm, John Bashinski, Steve Rich, Jim Bieda, Dan Wing, Joe Clarke, Henk Birkholz, Adam Montville, and Robert Sparks for their valuable advice and reviews. Russ Housley entirely rewrote Section 10 to be a complete module. Adrian Farrel provided the basis for privacy considerations text. Kent Watsen provided a thorough review of the architecture and the YANG model. The remaining errors in this work are entirely the responsibility of the authors.

18. References

18.1. Normative References

[I-D.ietf-netmod-acl-model]
Jethanandani, M., Huang, L., Agarwal, S., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-18 (work in progress), March 2018.

[I-D.ietf-netmod-entity]
Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", draft-ietf-netmod-entity-08 (work in progress), January 2018.

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

[IEEE8021AB]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks--Station and Media Access Control Connectivity Discovery", n.d..

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[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>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <https://www.rfc-editor.org/info/rfc2131>.

[RFC2618] Aboba, B. and G. Zorn, "RADIUS Authentication Client MIB", RFC 2618, DOI 10.17487/RFC2618, June 1999, <https://www.rfc-editor.org/info/rfc2618>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <https://www.rfc-editor.org/info/rfc3315>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <https://www.rfc-editor.org/info/rfc3987>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <https://www.rfc-editor.org/info/rfc7227>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7610] Gont, F., Liu, W., and G. Van de Velde, "DHCPv6-Shield: Protecting against Rogue DHCPv6 Servers", BCP 199, RFC 7610, DOI 10.17487/RFC7610, August 2015, <https://www.rfc-editor.org/info/rfc7610>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

18.2. Informative References

[FW95] Chapman, D. and E. Zwicky, "Building Internet Firewalls", January 1995.

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-20 (work in progress), March 2018.

[IEEE8021AR]
Institute for Electrical and Electronics Engineers, "Secure Device Identity", 1998.

[ISO.8601.1988]
International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, June 1988.

[RFC1984] IAB and IESG, "IAB and IESG Statement on Cryptographic Technology and the Internet", BCP 200, RFC 1984, DOI 10.17487/RFC1984, August 1996, <https://www.rfc-editor.org/info/rfc1984>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <https://www.rfc-editor.org/info/rfc3553>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <https://www.rfc-editor.org/info/rfc6092>.

[RFC6872] Gurbani, V., Ed., Burger, E., Ed., Anjali, T., Abdelnur, H., and O. Festor, "The Common Log Format (CLF) for the Session Initiation Protocol (SIP): Framework and Information Model", RFC 6872, DOI 10.17487/RFC6872, February 2013, <https://www.rfc-editor.org/info/rfc6872>.

[RFC7042] Eastlake 3rd, D. and J. Abley, "IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters", BCP 141, RFC 7042, DOI 10.17487/RFC7042, October 2013, <https://www.rfc-editor.org/info/rfc7042>.

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <https://www.rfc-editor.org/info/rfc7170>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[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>.

[RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <https://www.rfc-editor.org/info/rfc7452>.

[RFC7488] Boucadair, M., Penno, R., Wing, D., Patil, P., and T. Reddy, "Port Control Protocol (PCP) Server Selection", RFC 7488, DOI 10.17487/RFC7488, March 2015, <https://www.rfc-editor.org/info/rfc7488>.

Appendix A. Changes from Earlier Versions

RFC Editor to remove this section prior to publication.

Draft -19: * Edits after discussion with apps area to address reserved field for the future. * Correct systeminfo to be utf8. * Remove "hardware-rev" from list.

Draft -18: * Correct an error in the augment statement * Changes to the ACL model re ports.

Draft -17:

- One editorial.

Draft -16

- add mud-signature element based on review comments
- redo mud-url
- make clear that systeminfo uses UTF8

Draft -13 to -14:

- Final WGLC comments and review comments
- Move version from MUD-URL to Model

- Have MUD-URL in model
- Update based on update to draft-ietf-netmod-acl-model
- Point to tree diagram draft instead of 6087bis.

Draft -12 to -13:

- Additional WGLC comments

Draft -10 to -12:

These are based on WGLC comments:

- Correct examples based on ACL model changes.
- Change ordering nodes.
- Additional explanatory text around systeminfo.
- Change ordering in examples.
- Make it VERY VERY VERY VERY clear that these are recommendations, not mandates.
- DHCP -> NTP in some of the intro text.
- Remove masa-server
- "Things" to "network elements" in a few key places.
- Reference to JSON YANG RFC added.

Draft -10 to -11:

- Example corrections
- Typo
- Fix two lists.
- Addition of 'any-acl' and 'mud-acl' in the list of allowed features.
- Clarification of what should be in a MUD file.

Draft -09 to -10:

- AD input.
- Correct dates.
- Add compliance sentence as to which ACL module features are implemented.

Draft -08 to -09:

- Resolution of Security Area review, IoT directorate review, GenART review, YANG doctors review.
- change of YANG structure to address mandatory nodes.
- Terminology cleanup.
- specify out extra portion of MUD-URL.
- consistency changes.
- improved YANG descriptions.
- Remove extra revisions.
- Track ACL model changes.
- Additional cautions on use of ACL model; further clarifications on extensions.

Draft -07 to -08:

- a number of editorials corrected.
- definition of MUD file tweaked.

Draft -06 to -07:

- Examples updated.
- Additional clarification for direction-initiated.
- Additional implementation guidance given.

Draft -06 to -07:

- Update models to match new ACL model

- extract directionality from the ACL, introducing a new device container.

Draft -05 to -06:

- Make clear that this is a component architecture (Polk and Watson)
- Add order of operations (Watson)
- Add extensions leaf-list (Pritikin)
- Remove previous-mud-file (Watson)
- Modify text in last-update (Watson)
- Clarify local networks (Weis, Watson)
- Fix contact info (Watson)
- Terminology clarification (Weis)
- Advice on how to handle LDevIDs (Watson)
- Add deployment considerations (Watson)
- Add some additional text about fingerprinting (Watson)
- Appropriate references to 6087bis (Watson)
- Change systeminfo to a URL to be referenced (Lear)

Draft -04 to -05: * syntax error correction

Draft -03 to -04: * Re-add my-controller

Draft -02 to -03: * Additional IANA updates * Format correction in YANG. * Add reference to TEAP.

Draft -01 to -02: * Update IANA considerations * Accept Russ Housley rewrite of X.509 text * Include privacy considerations text * Redo the URL limit. Still 255 bytes, but now stated in the URL definition. * Change URI registration to be under urn:ietf:params

Draft -00 to -01: * Fix cert trust text. * change supportInformation to meta-info * Add an informational element in. * add urn registry and create first entry * add default elements

Appendix B. Default MUD nodes

What follows is the portion of a MUD file that permits DNS traffic to a controller that is registered with the URN "urn:ietf:params:mud:dns" and traffic NTP to a controller that is registered "urn:ietf:params:mud:ntp". This is considered the default behavior and the ACEs are in effect appended to whatever other "ace" entries that a MUD file contains. To block DNS or NTP one repeats the matching statement but replaces the "forwarding" action "accept" with "drop". Because ACEs are processed in the order they are received, the defaults would not be reached. A MUD controller might further decide to optimize to simply not include the defaults when they are overriden.

Four "acl" list entries that implement default MUD nodes are listed below. Two are for IPv4 and two are for IPv6 (one in each direction for both versions of IP). Note that neither access-list name nor ace name need be retained or used in any way by local implementations, but are simply there for completeness' sake.

```
 "ietf-access-control-list:access-lists": {
    "acl": [
      {
        "name": "mud-59776-v4to",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "ent0-todev",
              "matches": {
                "ietf-mud:mud": {
                  "controller": "urn:ietf:params:mud:dns"
                },
                "ipv4": {
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
                    "operator": "eq",
                    "port": 53
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            },
            {
```

```
                "name": "ent1-todev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "ipv4": {
                    "protocol": 17
                  },
                  "udp": {
                    "source-port": {
                      "operator": "eq",
                      "port": 123
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        },
        {
          "name": "mud-59776-v4fr",
          "type": "ipv4-acl-type",
          "aces": {
            "ace": [
              {
                "name": "ent0-frdev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "ipv4": {
                    "protocol": 17
                  },
                  "udp": {
                    "destination-port": {
                      "operator": "eq",
                      "port": 53
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              },
              {
```

```
                "name": "ent1-frdev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "ipv4": {
                    "protocol": 17
                  },
                  "udp": {
                    "destination-port": {
                      "operator": "eq",
                      "port": 123
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        },
        {
          "name": "mud-59776-v6to",
          "type": "ipv6-acl-type",
          "aces": {
            "ace": [
              {
                "name": "ent0-todev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "ipv6": {
                    "protocol": 17
                  },
                  "udp": {
                    "source-port": {
                      "operator": "eq",
                      "port": 53
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              },
              {
```

```
                "name": "ent1-todev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "ipv6": {
                    "protocol": 17
                  },
                  "udp": {
                    "source-port": {
                      "operator": "eq",
                      "port": 123
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        },
        {
          "name": "mud-59776-v6fr",
          "type": "ipv6-acl-type",
          "aces": {
            "ace": [
              {
                "name": "ent0-frdev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "ipv6": {
                    "protocol": 17
                  },
                  "udp": {
                    "destination-port": {
                      "operator": "eq",
                      "port": 53
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              },
              {
```

```
                "name": "ent1-frdev",
                "matches": {
                  "ietf-mud:mud": {
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "ipv6": {
                    "protocol": 17
                  },
                  "udp": {
                    "destination-port": {
                      "operator": "eq",
                      "port": 123
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        }
      ]
    }
```

Appendix C. A Sample Extension: DETNET-indicator

In this sample extension we augment the core MUD model to indicate whether the device implements DETNET. If a device later attempts to make use of DETNET, an notification or exception might be generated. Note that this example is intended only for illustrative purposes.

Extension Name: "Example-Extension" (to be used in the extensions list)
Standard: this document (but do not register the example)

This extension augments the MUD model to include a single node, using the following sample module that has the following tree structure:

```
module: ietf-mud-detext-example
  augment /ietf-mud:mud:
    +--rw is-detnet-required?   boolean
```

The model is defined as follows:

<CODE BEGINS>file "ietf-mud-detext-example@2018-03-01.yang"

```
   module ietf-mud-detext-example {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-mud-detext-example";
     prefix ietf-mud-detext-example;

     import ietf-mud {
       prefix ietf-mud;
     }

     organization
       "IETF OPSAWG (Ops Area) Working Group";
     contact
       "WG Web: http://tools.ietf.org/wg/opsawg/
        WG List: opsawg@ietf.org
        Author: Eliot Lear
        lear@cisco.com
        Author: Ralph Droms
        rdroms@gmail.com
        Author: Dan Romascanu
        dromasca@gmail.com

       ";
     description
       "Sample extension to a MUD module to indicate a need
        for DETNET support.";

     revision 2018-03-01 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: Manufacturer Usage Description
          Specification";
     }

     augment "/ietf-mud:mud" {
       description
         "This adds a simple extension for a manufacturer
           to indicate whether DETNET is required by a
          device.";
       leaf is-detnet-required {
         type boolean;
         description
           "This value will equal true if a device requires
            detnet to properly function";
       }
     }
   }
   <CODE ENDS>
```

Using the previous example, we now show how the extension would be expressed:

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://lighting.example.com/lightbulb2000",
    "last-update": "2018-03-02T11:20:51+01:00",
    "cache-validity": 48,
    "extensions": [
        "ietf-mud-detext-example"
     ],
    "ietf-mud-detext-example:is-detnet-required": "false",
    "is-supported": true,
    "systeminfo": "The BMS Example Lightbulb",
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6fr"
          }
        ]
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6to"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:access-lists": {
    "acl": [
      {
        "name": "mud-76100-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-todev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:src-dnsname": "test.com",
                  "protocol": 6
                },
```

```
                "tcp": {
                  "ietf-mud:direction-initiated": "from-device",
                  "source-port": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "mud-76100-v6fr",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-frdev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:dst-dnsname": "test.com",
                  "protocol": 6
                },
                "tcp": {
                  "ietf-mud:direction-initiated": "from-device",
                  "destination-port": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Ralph Droms
Google
355 Main St., 5th Floor
Cambridge

Phone: +1 978 376 3731
Email: rdroms@gmail.com

Dan Romascanu

Phone: +972 54 5555347
Email: dromasca@gmail.com