           Manufacturer Usage Description Specification
                     draft-ietf-opsawg-mud-25

Abstract

   This memo specifies a component-based architecture for manufacturer
   usage descriptions (MUD). The goal of MUD is to provide a means for
   end devices to signal to the network what sort of access and network
   functionality they require to properly function. 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 December 17, 2018.

Copyright Notice

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 intended to be used for general purpose computing
   tasks. These devices, which this memo refers to as Things, have a
   specific purpose. By definition, therefore, all other uses are not
   intended. If a small number of communication patterns follows from
   those small number of uses, 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. MUD primarily addresses

threats to the device rather than the device as a threat. In some
circumstances, however, MUD may offer some protection in the latter
case, depending on the MUD-URL is communicated, and how devices and
their communications are authenticated.

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 exists 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:

o Substantially reduce the threat surface on a device to those
   communications intended by the manufacturer.

o Provide a means to scale network policies to the ever-increasing
   number of types of devices in the network.

o 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.

o Keep the cost of implementation of such a system to the bare
   minimum.

o Provide a means of extensibility for manufacturers to express
   other device capabilities or requirements.

MUD consists of three architectural building blocks:

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

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

o A means for local network management systems to retrieve the
   description.

MUD is most effective when the network is able to identify in some
way the remote endpoints that Things will talk to.

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 manager: 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 controller: a synonym that has been used in the past for MUD
    manager.

MUD URL: a URL that can be used by the MUD manager 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", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in BCP
14 [RFC2119] [RFC8174] when, and only when, they appear in all
capitals, as shown here.

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
solution, however, we assume that a device will implement
functionality necessary to fulfill its limited purpose. This is
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:

o A DHCP option[RFC2131],[RFC3315] that the DHCP client uses to
   inform the DHCP server. The DHCP server may take further actions,
   such as act as the MUD manager or otherwise pass the MUD URL along
   to the MUD manager.

o 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].

o 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 are encouraged to allow for this sort of
flexibility of how MUD URLs may be learned.

1.6. Processing of the MUD URL

MUD managers 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 [RFC2818],
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 field ([RFC7231], Section 5.3.5), and a "User-Agent"
header ([RFC7231], Section 5.5.3).

MUD managers SHOULD automatically process 3xx response status codes.

If a MUD manager 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.

It may not be possible for a MUD manager to retrieve a MUD file at
any given time. Should a MUD manager 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 manager 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 interfere with the deployment of a device. It is a
local decision as to how to handle such circumstances.

1.7. Types of Policies

When the MUD URL is resolved, the MUD manager 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:

o Allow access to devices of the same manufacturer

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

o Allow access to local DNS/NTP

o Deny all other access

A printer might have a description that states:

o Allow access for port IPP or port LPD

o Allow local access for port HTTP

o 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 it provides accurate and adequate
models for use by network devices. JSON[RFC8259] is used as a

serialization format 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 intended to serve as controllers for the MUD-
   URL that the Thing emitted.

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 the YANG-based
configuration in a MUD file is limited to either the modules
specified or referenced in this document, or those specified in
documented extensions.

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.

```
     ......................................
     .                    _____   .         _____
     .                   |            |  .        |             |
     .                   |    MUD     |-->get URL-->|    MUD    |
     .                   |  Manager   |  .(https) | File Server |
     .  End system network |_____|<-MUD file<-<|_____|
     .                            .          .
     .                            .          .
     . _____             _____     .
     .|        |  (dhcp et al) |  router  |  .
     .|  Thing |---->MUD URL-->|   or     |  .
     .|_____|            |  switch  |  .
     .                      |_____|  .
     ......................................
```
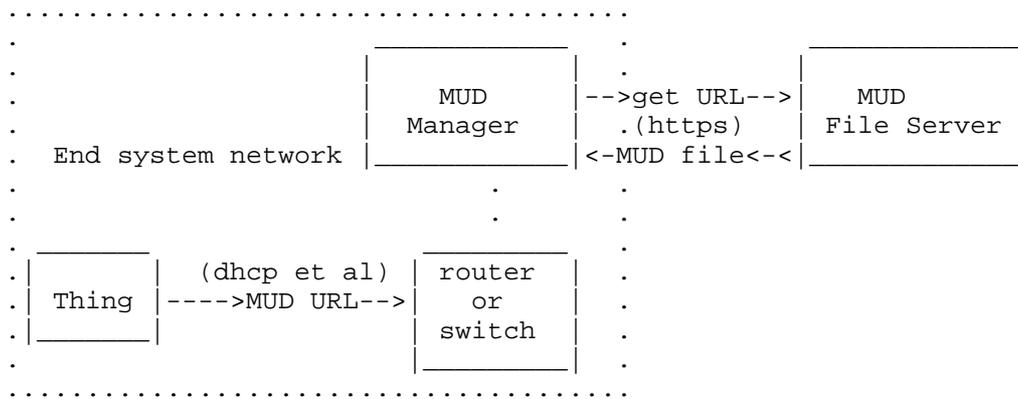
                    Figure 1: MUD Architecture

   In the above diagram, the switch or router collects MUD URLs and
   forwards them to the MUD manager (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 [IEEE8021X], 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 is valid for as long
   as the Thing is connected. There is no expiry. However, if the MUD
   manager has detected that the MUD file for a Thing has changed, it
   SHOULD update the policy expeditiously, taking into account whatever
   approval flow is required in a deployment. In this way, new
   recommendations from the manufacturer can be processed in a timely
   fashion.

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. In the trivial case
it may hardcode MUD-URL on a switch port or a map from some available
identifier such as an L2 address or certificate hash to a MUD-URL.

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

o receive MUD URLs,

o fetch MUD files,

o translate abstractions in the MUD files to specific network
   element configuration,

o maintain and update any required mappings of the abstractions, and

o update network elements with appropriate configuration.

A MUD manager 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 manager by the nearest switch (how
    this happens depends on the way in which the MUD URL is emitted).

3. The MUD manager 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 manager 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 manager instantiates local configuration based on the
    abstractions defined in this document.

6. The MUD manager 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 instance 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:

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

   o ietf-mud (this document)

   o 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 manager 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:

   o 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 manager MAY choose to interpret "reject" as "drop". A MUD
   manager 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 [RFC8343] 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 managers 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 3.9. See that
section for more information.

2.1. The IETF-MUD YANG Module

This module is structured into three parts:

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

o 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.

o 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 acls, 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 [RFC8340].

```
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 documentation?       inet:uri
     +--rw extensions*          string
     +--rw from-device-policy
     |  +--rw acls
     |     +--rw access-list* [name]
     |        +--rw name     -> /acl:acls/acl/name
     +--rw to-device-policy
        +--rw acls
           +--rw access-list* [name]
              +--rw name     -> /acl:acls/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:acls/acl:acl/acl:aces/acl:ace/acl:matches
       /acl:l4/acl:tcp/acl:tcp:
    +--rw direction-initiated?   direction
```

3. MUD model definitions for the root mud container

3.1. mud-version

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

3.2. mud-url

   This URL identifies the MUD file. This is useful when the file and
   associated signature are manually uploaded, say, in an offline mode.

3.3. 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.

3.4. 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].

3.5. 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 manager to discard
   the MUD file, nor terminate access to a Thing. See Section 16 for
   more information.

3.6. 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 a firmware or software update to the Thing or
   never update the MUD file. A MUD manager MAY still periodically
   check for updates.

3.7. 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 brief

displayable description of the Thing. It SHOULD NOT exceed 60
characters worth of display space.

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

These optional fields are filled in as specified by [RFC8348]. 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 MUD-URLs that are contained in
802.1AR certificates.

3.9. extensions

This optional leaf-list names MUD extensions that are used in the MUD
file. Note that MUD extensions MUST NOT 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.

4. Augmentation to the ACL Model

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

4.1. 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.

4.2. 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.

4.3. documentation

   This URI consists of a URL that points to documentation relating to
   the device and the MUD file. This can prove particularly useful when
   the "controller" class is used, so that its use can be explained.

4.4. 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.

4.5. 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.

4.6. controller

   This URI specifies a value that a controller will register with the
   MUD manager. 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, that has been
   standardized. Both of those URNs may be found in Section 17.6.

   When "my-controller" is used, it is possible that the administrator
   will be prompted to populate that class for each and every model.
   Use of "controller" with a named class allows the user to populate
   that class only once for many different models that a manufacturer
   may produce.

   Controller URIs MAY take the form of a URL (e.g. "http[s]://").
   However, MUD managers 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.

   Great care should be taken by MUD managers when invoking the
   controller class in the form of URLs. For one thing, it requires
   some understanding by the administrator as to when it is appropriate.
   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.

4.7. my-controller

This null-valued node signals to the MUD manager 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.

4.8. direction-initiated

This MUST only be applied to TCP. This matches the direction in which a TCP connection is initiated. When direction initiated is "from-device", packets that are transmitted in the direction of a thing MUST be dropped unless the thing has first initiated a TCP connection. By way of example, this node may be implemented in its simplest form by looking at naked SYN bits, but may also be implemented through more stateful mechanisms.

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.

5. Processing of the MUD file

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

o Anything not explicitly permitted is denied.

o 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.

6. 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
```

   A manufacturer may construct a MUD URL in any way, so long as it
   makes use of the "https" schema.

7. The MUD YANG Model

```
   <CODE BEGINS>file "ietf-mud@2018-06-15.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 2018-06-15 {
    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 Managers 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 manager 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 documentation {
        type inet:uri;
        description
          "This URL points to documentation that
           relates to this device and any classes that it uses
           in its MUD file.  A caution: MUD managers need
           not resolve this URL on their own, but rather simply
           provide it to the administrator.  Parsing HTML is
           not an intended function of a MUD manager.";
      }
      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 enforcement 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 enforcement 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>
```

8. 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:acls/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:acls/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.

8.1. 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.

8.2. 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.

8.3. The ietf-acldns Model

```
<CODE BEGINS>file "ietf-acldns@2018-06-15.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 2018-06-15 {
       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>
```

9. 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:acls": {
    "acl": [
      {
        "name": "mud-76100-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-todev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:src-dnsname": "test.example.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.example.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.

10. The MUD URL DHCP Option

The IPv4 MUD URL client option has the following format:

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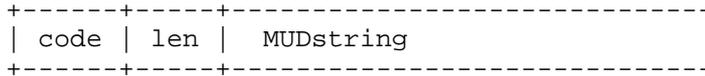
Code OPTION_MUD_URL_V4 (161) is assigned by IANA. len is a single
octet that indicates the length of MUD string in octets. The MUD
string is defined as follows:

```
 MUDstring = mudurl [ " " reserved ]
 mudurl = URI; a URL [RFC3986] that uses the "https" schema [RFC7230]
 reserved = 1*( OCTET ) ; 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 managers that do not understand this
field MUST ignore it.

The IPv6 MUD URL client option has the following format:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          OPTION_MUD_URL_V6     |          option-length        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          MUDstring                            |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
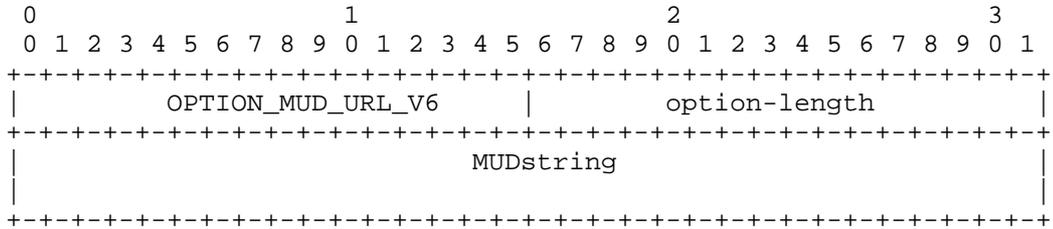
OPTION_MUD_URL_V6 (112; assigned by IANA).

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.

10.1. 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.

10.2. Server Behavior

   A DHCP server may ignore these options or take action based on
   receipt of these options. When a server consumes this option, it
   will either forward the URL and relevant client information (such as
   the gateway address or giaddr and requested IP address, and lease
   length) 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 manager. 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 manager MUST notify the MUD manager of any
   corresponding change to the DHCP state of the client (such as
   expiration or explicit release of a network address lease).

   Should the DHCP server fail, in the case when it implements the MUD
   manager functionality, any backup mechanisms SHOULD include the MUD
   state, and the server SHOULD resolve the status of clients upon its
   restart, similar to what it would do, absent MUD manager
   functionality. In the case where the DHCP server forwards
   information to the MUD manager, the MUD manager will either make use
   of redundant DHCP servers for information, or otherwise clear state
   based on other network information, such as monitoring port status on
   a switch via SNMP, Radius accounting, or similar mechanisms.

10.3. Relay Requirements

   There are no additional requirements for relays.

11. 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 6 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.

   In addition, a separate new extension is defined as id-pe-mudsigner.
   This contains the subject field of the signing certificate of the MUD
   file. Processing of this field is specified in Section 13.2.

   The purpose of this signature is to make a claim that the MUD file
   found on the server is valid for a given device, independent of any
   other factors. There are several security considerations below in
   Section 16.

   A new content-type id-ct-mud is also defined. While signatures are
   detached today, should a MUD file be transmitted as part of a CMS
   message, this content-type SHOULD be used.

   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

   -- RFC 5912
   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) }

   -- RFC 5912
   id-ct
   FROM PKIXCRMF-2009
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5)  mechanisms(5) pkix(7) id-mod(0)
         id-mod-crmf2005-02(55) }

   -- RFC 6268
   CONTENT-TYPE
   FROM CryptographicMessageSyntax-2010
     { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

   -- RFC 5912
   id-pe, Name
   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) } ;

 --
 -- Certificate Extensions
 --

 MUDCertExtensions EXTENSION ::=
    { ext-MUDURL | ext-MUDsigner, ... }

 ext-MUDURL EXTENSION ::=
    { SYNTAX MUDURLSyntax IDENTIFIED BY id-pe-mud-url }

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

 MUDURLSyntax ::= IA5String

 ext-MUDsigner EXTENSION ::=
    { SYNTAX MUDsignerSyntax IDENTIFIED BY id-pe-mudsigner }
```

```
      id-pe-mudsigner OBJECT IDENTIFIER ::= { id-pe TBD1 }

      MUDsignerSyntax ::= Name


      --
      -- CMS Content Types
      --

      MUDContentTypes CONTENT-TYPE ::=
         { ct-mud, ... }

       ct-mud CONTENT-TYPE ::=
          { -- directly include the content
            IDENTIFIED BY id-ct-mudtype }
          -- The binary data that is in the form
          -- 'application/mud+json" is directly encoded as the
          -- signed data.  No additional ASN.1 encoding is added.

      id-ct-mudtype OBJECT IDENTIFIER ::= { id-ct TBD2 }

      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 manager implementations maintain a table
   that maps a Thing to its MUD URL based on IDevIDs.

12. 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:

```
      +--------+--------+----------+---------+--------------
      |TLV Type|  len   |   OUI    |subtype  |  MUDString
      |  =127  |        |= 00 00 5E|  = 1    |
      |(7 bits)|(9 bits)|(3 octets)|(1 octet)|(1-255 octets)
      +--------+--------+----------+---------+--------------
```

where:

o TLV Type = 127 indicates a vendor-specific TLV

o len - indicates the TLV string length

o OUI = 00 00 5E is the organizationally unique identifier of IANA

o subtype = 1 (to be assigned by IANA for the MUD URL)

o 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 manager, or it will retrieve the
usage description by resolving the URL in accordance with normal HTTP
semantics.

13. 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 ensure 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.

13.1. 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 -binary \
              -certfile intermediatecert -out mudfile.p7s

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

13.2. Verifying a MUD file signature

   Prior to processing the rest of a MUD file, the MUD manager MUST
   retrieve the MUD signature file by retrieving the value of "mud-
   signature" and validating the signature across the MUD file. The Key
   Usage Extension in the signing certificate MUST be present and have
   the bit digitalSignature(0) set. When the id-pe-mudsigner extension
   is present in a device's X.509 certificate, the MUD signature file
   MUST have been generated by a certificate whose subject matches the
   contents of that id-pe-mudsigner extension. If these conditions are
   not met, or if it cannot validate the chain of trust to a known trust
   anchor, the MUD manager MUST cease processing the MUD file 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 just the
   relationship between the Thing 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.

14. 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 managers 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.

15. 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 lateral infection (infection of
   devices that reside near one another) 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 allowed 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.

Finally, please see directly below regarding device lifetimes and use
of domain names.

16. 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. This can happen
in a number of ways when a device emits a MUD URL using DHCP or LLDP,
such as being inappropriately admitted to a class such as "same-
manufacturer", given access to a device such as "my-controller", or
being permitted access to an Internet resource, where such access
would otherwise be disallowed. Whether that is the case will depend
on the deployment. Implementations SHOULD be configurable to
disallow additive access for devices using MUD-URLs that are not
emitted in a secure fashion such as in a certificate. Similarly,
implementations SHOULD NOT grant elevated permissions (beyond those
of devices presenting no MUD policy) to devices which do not strongly
bind their identity to their L2/L3 transmissions. When insecure
methods are used by the MUD Manager, the classes SHOULD NOT contain
devices that use both insecure and secure methods, in order to
prevent privilege escalation attacks, and MUST NOT contain devices
with the same MUD-URL that are derived from both strong and weak
authentication methods.

Devices may forge source (L2/L3) information. Deployments should
apply appropriate protections to bind communications to the
authentication that has taken place. For 802.1X authentication, IEEE
802.1AE (MACsec) [IEEE8021AE] is one means by which this may happen.
A similar approach can be used with 802.11i (WPA2) [IEEE80211i].
Other means are available with other lower layer technologies.
Implementations using session-oriented access that is not
cryptographically bound should take care to remove state when any
form of break in the session is detected.

A rogue CA may sign a certificate that contains the same subject name
as is listed in the MUDsigner field in the manufacturer certificate,
thus seemingly permitting a substitute MUD file for a device. There
are two mitigations available: first, if the signer changes, this may
be flagged as an exception by the MUD manager. If the MUD file also
changes, the MUD manager SHOULD seek administrator approval (it

should do this in any case). In all circumstances, the MUD manager
MUST maintain a cache of trusted CAs for this purpose. When such a
rogue is discovered, it SHOULD be removed.

Additional mitigations are described below.

When 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 be relevant
whenthe MUD manager 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 artifacts to the MUD
string in the form of the reserved field discussed in Section 10.
The meaning of such artifacts is left as future work.

MUD managers SHOULD NOT accept a usage description for a Thing with
the same MAC address that has indicated a change of the URL 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
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 that the signer of the MUD file is known to and
trusted by the MUD manager. 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 managers would consider valid. There are
a few approaches that can mitigate this attack. First, MUD managers
SHOULD cache certificates used by the MUD file server. When a new
certificate is retrieved for whatever reason, the MUD manager 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 MUD
manager MAY check the WHOIS database to see if registration ownership

of a domain has changed. If a change has occurred, 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 manager has been
installed.

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].

There is the risk of the MUD manager itself being spied on to
determine what things are connected to the network. To address this
risk, MUD managers may choose to make use of TLS proxies that they
trust that would aggregate other information.

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.

17. IANA Considerations

[There was originally a registry entry for .well-known suffixes.
This has been removed from the draft and may be marked as deprecated
in the registry. RFC Editor: please remove this comment.]

17.1. YANG Module Registrations

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

   The ietf-mud module:

   o Name: ietf-mud

   o URN: urn:ietf:params:xml:ns:yang:ietf-mud

   o Prefix: ietf-mud

   o Registrant conact: The IESG

   o Reference: [RFCXXXX]

   The ietf-acldns module:

   o Name: ietf-acldns

   o URI: urn:ietf:params:xml:ns:yang:ietf-acldns

   o Prefix: ietf-acldns

   o Registrant: the IESG

   o Reference: [RFCXXXX]

17.2. 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 10.

17.3. 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).

   o id-pe-mudsigner object identifier from the "SMI Security for PKIX
   Certificate Extension" registry (TBD1).

   o id-ct-mudtype object identifier from the "SMI Security for S/MIME
   CMS Content Type" registry (TBD2).

   The use of these values is specified in Section 11.

17.4. MIME Media-type Registration for MUD files

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

   o Type name: application
   o Subtype name: mud+json
   o Required parameters: n/a
   o Optional parameters: n/a
   o Encoding considerations: 8bit; application/mud+json values
     are represented as a JSON object; UTF-8 encoding MUST be
     employed. [RFC3629]
   o Security considerations: See Security Considerations
     of RFCXXXX and [RFC8259] Section 12.
   o Interoperability considerations: n/a
   o Published specification: [RFCXXXX]
   o Applications that use this media type: MUD managers as
     specified by [RFCXXXX].
   o Fragment identifier considerations: n/a
   o Additional information:

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

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

17.5. 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 >

17.6. 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].

17.7. 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].

18. 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, Jim Schaad, and Robert Sparks for their
valuable advice and reviews. Russ Housley entirely rewrote

   Section 11 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.

19. References

19.1. 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-19 (work in progress), April
              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>.

   [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>.

   [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO
              10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November
              2003, <https://www.rfc-editor.org/info/rfc3629>.

   [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>.

   [RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for
              Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911,
              DOI 10.17487/RFC5911, June 2010,
              <https://www.rfc-editor.org/info/rfc5911>.

   [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the
              Public Key Infrastructure Using X.509 (PKIX)", RFC 5912,
              DOI 10.17487/RFC5912, June 2010,
              <https://www.rfc-editor.org/info/rfc5912>.

   [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>.

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

   [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data
              Interchange Format", STD 90, RFC 8259,
              DOI 10.17487/RFC8259, December 2017,
              <https://www.rfc-editor.org/info/rfc8259>.

   [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
              BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
              <https://www.rfc-editor.org/info/rfc8340>.

   [RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A
              YANG Data Model for Hardware Management", RFC 8348,
              DOI 10.17487/RFC8348, March 2018,
              <https://www.rfc-editor.org/info/rfc8348>.

19.2. 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.

   [IEEE80211i]
              Institute for Electrical and Electronics Engineers, "IEEE
              Standard for information technology-Telecommunications and
              information exchange between systems-Local and
              metropolitan area networks-Specific requirements-Part 11-
              Wireless LAN Medium Access Control (MAC) and Physical
              Layer (PHY) specifications- Amendment 6- Medium Access
              Control (MAC) Security Enhancements", 2004.

   [IEEE8021AE]
              Institute for Electrical and Electronics Engineers, "IEEE
              Standard for Local and Metropolitan Area Networks- Media
              Access Control (MAC) Security", 2006.

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

   [IEEE8021X]
              Institute for Electrical and Electronics Engineers, "IEEE
              Standard for Local and metropolitan area networks--Port-
              Based Network Access Control", 2010.

[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>.

[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>.

   [RFC8343] Bjorklund, M., "A YANG Data Model for Interface
              Management", RFC 8343, DOI 10.17487/RFC8343, March 2018,
              <https://www.rfc-editor.org/info/rfc8343>.

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:

   o One editorial.

   Draft -16

   o add mud-signature element based on review comments

   o redo mud-url

   o make clear that systeminfo uses UTF8

   Draft -13 to -14:

   o Final WGLC comments and review comments

   o Move version from MUD-URL to Model

   o Have MUD-URL in model

   o Update based on update to draft-ietf-netmod-acl-model

   o Point to tree diagram draft instead of 6087bis.

Draft -12 to -13:

o Additional WGLC comments

Draft -10 to -12:

These are based on WGLC comments:

o Correct examples based on ACL model changes.

o Change ordering nodes.

o Additional explanatory text around systeminfo.

o Change ordering in examples.

o Make it VERY VERY VERY VERY clear that these are recommendations,
   not mandates.

o DHCP -> NTP in some of the intro text.

o Remove masa-server

o "Things" to "network elements" in a few key places.

o Reference to JSON YANG RFC added.

Draft -10 to -11:

o Example corrections

o Typo

o Fix two lists.

o Addition of 'any-acl' and 'mud-acl' in the list of allowed
   features.

o Clarification of what should be in a MUD file.

Draft -09 to -10:

o AD input.

o Correct dates.

o Add compliance sentence as to which ACL module features are
   implemented.

Draft -08 to -09:

o Resolution of Security Area review, IoT directorate review, GenART
   review, YANG doctors review.

o change of YANG structure to address mandatory nodes.

o Terminology cleanup.

o specify out extra portion of MUD-URL.

o consistency changes.

o improved YANG descriptions.

o Remove extra revisions.

o Track ACL model changes.

o Additional cautions on use of ACL model; further clarifications on
   extensions.

Draft -07 to -08:

o a number of editorials corrected.

o definition of MUD file tweaked.

Draft -06 to -07:

o Examples updated.

o Additional clarification for direction-initiated.

o Additional implementation guidance given.

Draft -06 to -07:

o Update models to match new ACL model

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

Draft -05 to -06:

o Make clear that this is a component architecture (Polk and Watson)

o Add order of operations (Watson)

o Add extensions leaf-list (Pritikin)

o Remove previous-mud-file (Watson)

o Modify text in last-update (Watson)

o Clarify local networks (Weis, Watson)

o Fix contact info (Watson)

o Terminology clarification (Weis)

o Advice on how to handle LDevIDs (Watson)

o Add deployment considerations (Watson)

o Add some additional text about fingerprinting (Watson)

o Appropriate references to 6087bis (Watson)

o 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 manager might
further decide to optimize to simply not include the defaults when
they are overridden.

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:acls": {
    "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 claims not to use
   DETNET, but then later attempts to do so, a 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@2018-06-15.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 2018-06-15 {
     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:acls": {
      "acl": [
        {
          "name": "mud-76100-v6to",
          "type": "ipv6-acl-type",
          "aces": {
            "ace": [
              {
                "name": "cl0-todev",
                "matches": {
                  "ipv6": {
                    "ietf-acldns:src-dnsname": "test.example.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.example.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