Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 10, 2018

E. Lear
Cisco Systems
R. Droms

D. Romascanu
August 09, 2017

Manufacturer Usage Description Specification
draft-ietf-opsawg-mud-08

Abstract

This memo specifies a component-based architecture for manufacturer usage descriptions (MUD). This includes two YANG modules, IPv4 and IPv6 DHCP options, an LLDP TLV, a URL suffix specification, 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 http://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 February 10, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. What MUD doesn't do 4
1.2. A Simple Example 4
1.3. Determining Intended Use 5
1.4. Finding A Policy: The MUD URL 5
1.5. Types of Policies 6
1.6. Terminology 7
1.7. The Manufacturer Usage Description Architecture 8
1.8. Order of operations 9
2. The MUD Model and Semantic Meaning 10
3. Element Definitions 12
3.1. last-update 12
3.2. cache-validity 12
3.3. masa-server 12
3.4. is-supported 12
3.5. systeminfo 12
3.6. extensions 12
3.7. packet-direction 13
3.8. manufacturer 13
3.9. same-manufacturer 13
3.10. model 13
3.11. local-networks 13
3.12. controller 13
3.13. my-controller 14
3.14. direction-initiated 14
4. Processing of the MUD file 14
5. What does a MUD URL look like? 14
6. The MUD YANG Model 15
7. The Domain Name Extension to the ACL Model 21
7.1. source-dnsname 21
7.2. destination-dnsname 21
7.3. The ietf-acldns Model 21
8. MUD File Example 23
9. The MUD URL DHCP Option 25
9.1. Client Behavior 26
9.2. Server Behavior 26
9.3. Relay Requirements 27
10. The Manufacturer Usage Description (MUD) URL X.509 Extension 27
11. The Manufacturer Usage Description LLDP extension 28
12. Creating and Processing of Signed MUD Files 30
12.1. Creating a MUD file signature 30
12.2. Verifying a MUD file signature 30
13. Extensibility 31

14. Deployment Considerations 31
15. Security Considerations 32
16. IANA Considerations 33
16.1. YANG Module Registrations 33
16.2. DHCPv4 and DHCPv6 Options 34
16.3. PKIX Extensions 34
16.4. Well Known URI Suffix 34
16.5. MIME Media-type Registration for MUD files 34
16.6. LLDP IANA TLV Subtype Registry 35
16.7. The MUD Well Known Universal Resource Name (URNs) . . . 36
17. Acknowledgments 36
18. References 36
18.1. Normative References 36
18.2. Informative References 39
Appendix A. Changes from Earlier Versions 40
Appendix B. Default MUD elements 41
Authors' Addresses 47

1. Introduction

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

[RFC7452] discusses design patterns for, and poses questions about, smart objects. Let us then posit a group of objects that are specifically NOT general purpose computers. These devices therefore have a purpose to their use. By definition, therefore, all other purposes are NOT intended. The combination of these two statements can be restated as a manufacturer usage description (MUD) that can be applied at various points within a network. Although this memo may seem to stress access requirements, usage intent also consists of quality of service needs a device may have.

We use the notion of "manufacturer" loosely in this context, to simply mean the entity or organization that will state how a device is intended to be used. 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 integrator of that device. The key points are that the device itself is expected to serve a limited purpose, and that there may exist an organization in the supply chain of that device that will take responsibility for informing the network about that purpose.

The intent of MUD is to therefore solve for the following problems:

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

MUD therefore consists of three architectural building blocks: - A classifier that a device emits that can be used to locate a description; - The description itself, including how it is interpreted, and; - A means to retrieve the description.

In this specification we specify 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 enterprise networks for their own purposes.

1.1. What MUD doesn't do

MUD is not intended to address network authorization of general computing, 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.

No matter how good a MUD-enabled network is, it will never replace the need for manufacturers to patch vulnerabilities. It may, however, provide network administrators with some additional protection when those vulnerabilities exist.

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 app on smart phone accesses. 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 Facebook 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. 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]. Programmatically profiling systems have existed for years as well. These systems make use of heuristics that take at least some time to assert what a system is.

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

1.4. Finding A Policy: The MUD URL

Our work begins, therefore, 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.

In this memo three means are defined to emit the MUD URL. One is a DHCP option[RFC2131],[RFC3315] that the DHCP client uses to inform the DHCP server. The DHCP server may take further actions, such as retrieve the URL or otherwise pass it along to network management system or controller. The other method defined is an X.509 constraint. The IEEE has developed [IEEE8021AR] that provides a certificate-based approach to communicate device characteristics, which itself relies on [RFC5280]. The MUD URL extension is non-critical, as required by IEEE 802.1AR. Various means may be used to communicate that certificate, including Tunnel Extensible Authentication Protocol (TEAP) [RFC7170]. Finally, a Link Layer Discovery Protocol (LLDP) frame is defined [IEEE8021AB].

It is possible that there may be other means for a MUD URL to be learned by a network. For instance, if a device has a serial number, it may be possible for the MUD controller to perform a lookup of the device, if it has some knowledge as to who the device manufacturer

is, and what its MUD file server is. Such mechanisms are not described in this memo, but are possible.

1.5. Types of Policies

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

```
Allow access to devices of the same manufacturer
Allow access to and from controllers who need to speak COAP
Allow access to local DNS/DHCP
Deny all other access
```

To add a bit more depth that should not be a stretch of anyone's imagination, one could also make use of port-based access lists. Thus a printer might have a description that states:

```
Allow access for port IPP or port LPD
Allow local access for port HTTP
Deny all other access
```

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

The files that are retrieved are intended to be closely aligned to existing network architectures so that they are easy to deploy. We make use of YANG [RFC6020] because of the time and effort spent to develop accurate and adequate models for use by network devices. JSON is used as a serialization for compactness and readability, relative to XML.

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, and by leveraging YANG-based models, we provide the opportunity for new policies to be introduced as required.

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 are 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: A device that the local network administrator admits to the particular class.

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

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 therefore, mature. Therefore, only a a limited subset YANG-based configuration of is permitted in a MUD file.

1.6. Terminology

MUD: manufacturer usage description.

MUD file: a file containing YANG-based JSON that describes a recommended behavior.

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

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

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

Thing: the end device that emits 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 device. It could, for instance, be a systems integrator, or even a component provider.

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

1.7. The Manufacturer Usage Description Architecture

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

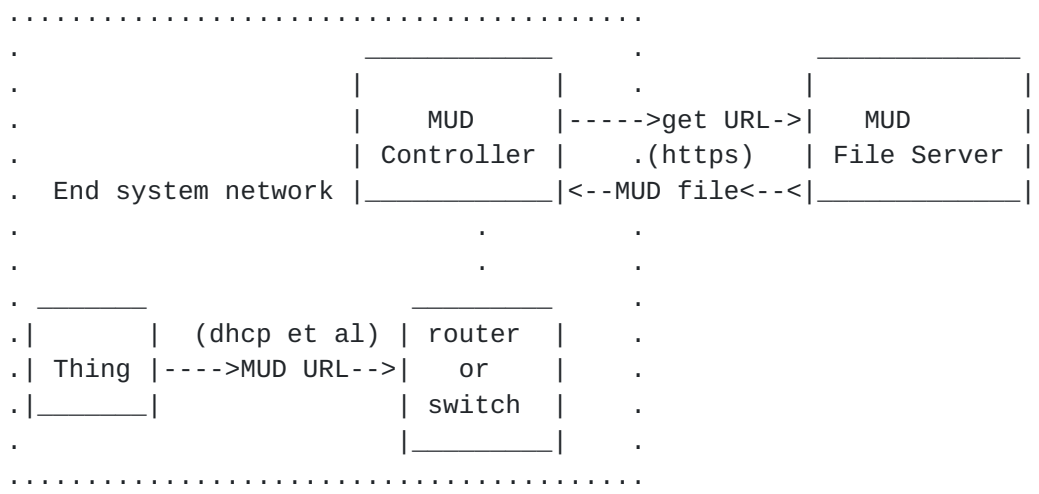


Figure 1: MUD Architecture

In the above diagram, the switch or router collects MUD URLs and forwards them to the network management system for processing. This happens in different ways, depending on how the URI is communicated. For instance, in the case of DHCP, the DHCP server might receive the URI and then process it. In the case of IEEE 802.1X, the switch would carry the URI 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 web site is valid for the duration of the device's connection, or as specified in the description. Thus if the device is mobile, when it moves on, any configuration in the switch is removed. Similarly, from time to time the description may be refreshed, based on new capabilities or communication patterns or vulnerabilities.

The web site 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 URI, it may proxy it, the trivial cases being a map between some registered device or port and a URL.

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

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

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

1.8. 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. Device emits URL.
2. That URL is forwarded to a MUD controller by the nearest switch (how this happens depends on the way in which the MUD URL is emitted).
3. The MUD controller retrieves the MUD file from the MUD file server, assuming it doesn't already have a copy. It may test the URL against a reputation service, and it may test any hosts within the file against reputation services, as it deems fit.

4. The MUD controller may query the administrator for permission to add the device and associated policy. If the device is known or the device type is known, it may skip this step.

5. The MUD controller instantiates local configuration based on the abstractions defined in this document.

6. The MUD controller configures the switch nearest the device. Other systems may be configured as well.

7. When the device disconnects, policy is removed.

2. The MUD Model and Semantic Meaning

A MUD file consists of JSON based on a YANG model. For purposes of MUD, the elements that can be modified are access lists as augmented by this model. The MUD file is limited to the serialization of a small number of YANG schema, including the models specified in the following documents:

- [I-D.ietf-netmod-acl-model]

- [RFC6991]

The ACL model makes use of features for different forms of ACLs. For purposes of MUD, any feature not related to addresses MAY be included. Any feature related to addresses MUST NOT be included. The reason for these statements is so that the broadest number of implementations will understand the the MUD file without need for negotiation. Furthermore, it is important to rely on appropriate abstractions stated in this memo. It is the job of the MUD controller to translate those abstractions to appropriate configuration for each network element.

Publishers of MUD files MUST NOT include other elements except as described in Section 3.6, and MUST only contain information relevant to the device being described. Devices parsing MUD files MUST cease processing if they find other elements.

======= This module is structured into four parts:

- The first container, metainfo, holds information that is relevant to retrieval and validity of the MUD file itself.

- The second container, device, describes the policies to be applied to traffic to and from the device. The only policy currently defined is a list of access control lists, but the from-device-

policy and to-device-policy containers serve as extension points for subsequent augmentation.

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

- The fourth container 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 simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is defined in [I-D.ietf-netmod-rfc6087bis].

```
module: ietf-mud
    +--rw metainfo
    |  +--rw last-update?      yang:date-and-time
    |  +--rw cache-validity?   uint8
    |  +--rw masa-server?      inet:uri
    |  +--rw is-supported?     boolean
    |  +--rw systeminfo?       inet:uri
    |  +--rw extensions*       string
    +--rw device
       +--rw from-device-policy
       |  +--rw access-lists
       |     +--rw access-list* [acl-name acl-type]
       |        +--rw acl-name    -> /acl:access-lists/acl/acl-name
       |        +--rw acl-type    identityref
       +--rw to-device-policy
          +--rw access-lists
             +--rw access-list* [acl-name acl-type]
                +--rw acl-name    -> /acl:access-lists/acl/acl-name
                +--rw acl-type    identityref
  augment /acl:access-lists/acl:acl/acl:access-list-entries \
          /acl:ace/acl:matches:
    +--rw mud-acl {mud-acl}?
       +--rw manufacturer?        inet:host
       +--rw same-manufacturer?   empty
       +--rw model?               string
       +--rw local-networks?      empty
       +--rw controller?          inet:uri
       +--rw my-controller?       empty
  augment /acl:access-lists/acl:acl/acl:access-list-entries \
          /acl:ace/acl:matches/acl:tcp-acl:
    +--rw direction-initiated?   direction
```

3. Element Definitions

The following elements are defined.

3.1. 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, turn was taken from Section 5.6 of [RFC3339], which was taken from [ISO.8601.1988].

3.2. 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 device that is supported.

3.3. masa-server

This optional element refers to the URL that should be used to resolve the location any MASA service, as specified in [I-D.ietf-anima-bootstrapping-keyinfra].

3.4. is-supported

This boolean is an indication from the manufacturer to the network administrator as to whether or not the device is supported. In this context a device is said to be supported if the manufacturer might issue an update to the device or if the manufacturer might update the MUD file.

3.5. systeminfo

This is a URL that points to a description of the device to be connected. The intent is for administrators to be able to read about what the device is the first time the MUD-URL is used.

3.6. extensions

This optional leaf-list names MUD extensions that are used in the MUD file. Note that NO MUD extensions may be used in a MUD file prior to the extensions being declared. Implementations MUST ignore any elements in this file that they do not understand.

3.7. packet-direction

[I-D.ietf-netmod-acl-model] describes access-lists but does not attempt to indicate where they are applied as that is handled elsewhere in a configuration. However, in this case, a MUD file must be explicit in describing the communication pattern of a device, and that includes indicating what is to be permitted or denied in either direction of communication. This element takes a single value of either "to-device" or "from-device", based on a typedef "direction". This is applied specifically for TCP, and may be implemented either via stateful or stateless approaches (e.g,. TCP flags)

3.8. manufacturer

This element consists of a hostname that would be matched against the authority component of another device'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.

3.9. same-manufacturer

This is an equivalent for when the manufacturer element is used to indicate the authority that is found in another device's MUD URL matches that of the authority found in this device's MUD URL.

3.10. model

This string matches the entire MUD URL, thus covering the model that is unique within the context of the authority. It may also include product version information. Thus how this field is constructed is entirely a local matter for the manufacturer.

3.11. local-networks

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

3.12. controller

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

Great care should be used when invoking the controller class. For one thing, it requires some understanding by the administrator as to when it is appropriate. Classes that are standardized may make it possible to code in certain intelligence. Nonstandard classes may require substantially more care. 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.

3.13. my-controller

This null-valued element establishes a class of controllers that are intended to control the device associated with a given MUD-URL.

3.14. direction-initiated

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

4. Processing of the MUD file

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

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

An explicit description of the defaults can be found in Appendix B.

5. What does a MUD URL look like?

To begin with, MUD takes full advantage of both the https: scheme and the use of .well-known. HTTPS is important in this case because a man in the middle attack could otherwise harm the operation of a class of devices. .well-known is used because we wish to add additional structure to the URL. And so the URL appears as follows:

```
    mud-url   = "https://" authority  "/.well-known/mud/" mud-rev
                "/" model ( "?" extras )
                ; authority is from RFC3986
    mud-rev   = "v1"
    model     = segment  ; from RFC3986
    extras    = query    ; from RFC3986
```

mud-rev signifies the version of the manufacturer usage description file. This memo specifies "v1" of that file. Later versions may permit additional schemas or modify the format. In order to provide for the broadest compatibility for the various transmission mechanisms, the length of the URL for v1 MUST NOT exceed 255 octets.

"model" represents a device model as the manufacturer wishes to represent it. It could be a brand name or something more specific. It also may provide a means to indicate what version the product is. Specifically if it has been updated in the field, this is the place where evidence of that update would appear. The field should be changed when the intended communication patterns of a device change. While from a controller standpoint, only comparison and matching operations are safe, it is envisioned that updates will require some administrative review. Processing of this URL occurs as specified in [RFC2818] and [RFC3986].

6. The MUD YANG Model

```
<CODE BEGINS>file "ietf-mud@2017-07-03.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.

     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 "2017-07-03"  {
     description
       "More work on ACL usage.";
     reference "RFC XXXX: Manufacturer Usage Description
                Specification";
   }

   revision "2017-06-29"  {
     description
       "Take packet directionality out of ACL. " +
       "Introduce device container.";
     reference "RFC XXXX: Manufacturer Usage Description
                Specification";
   }

   revision "2017-04-18"  {
     description "Base version of MUD extensions to ACL model";
     reference "RFC XXXX: Manufacturer Usage Description
                Specification";
```

```
   }

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

   identity mud-acl {
     base "acl:acl-base";
     description
       "ACL that contains MUD-specific access control list entries.";
   }

   feature mud-acl {
     description
       "MUD ACL augmentations supported.";
   }

   container metainfo  {

     description "Information about when support end(ed), and
                  when to refresh";

     leaf last-update {
       type yang:date-and-time;
       description "This is intended to be when
                    the MUD file was generated.";
     }

     leaf cache-validity {
       type uint8 {
         range "1..168";
       }
       description "The information retrieved from the MUD server is
                    valid for these many hours, after which it should
                    be refreshed.";
     }

     leaf masa-server {
```

```
      type inet:uri;
      description "The URI of the MASA server that network
      elements should forward requests to for this device.";
    }

    leaf is-supported {
      type boolean;
      description "The element is currently supported
                   by the manufacturer.";
    }
    leaf systeminfo {
      type inet:uri;
      description "A reference to a description of this device";
    }

    leaf-list extensions {
      type string;
      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.";
   }
  }

  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 "acl-name acl-type";
        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 acl-name {
          type leafref {
            path "/acl:access-lists/acl:acl/acl:acl-name";
          }
          description "The name of the ACL for this entry.";
        }
        leaf acl-type {
          type identityref {
            base acl:acl-base;
          }
          description "The type of the ACL for this entry.";
        }
      }
```

```
    }
  }

  container device {
    description "A container allowing us to associate data with the
                 device type being described in the instance document";

    container from-device-policy {
      description "The policies that should be enforced on traffic
                   coming from the device. These policies are not
                   necessarily intended to be enforced at a single
                   point, but may be rendered by the controller to any
                   relevant enorcement points in the network or
                   elsewhere.";
      uses access_lists;
    }

    container to-device-policy {
      description "The policies that should be enforced on traffic
                   going to the device. These policies are not
                   necessarily intended to be enforced at a single
                   point, but may be rendered by the controller to any
                   relevant enorcement points in the network or
                   elsewhere.";
      uses access_lists;
    }

  }

  augment "/acl:access-lists/acl:acl/" +
     "acl:access-list-entries/acl:ace/" +
     "acl:matches" {
    description "adding abstractions to avoid need of IP addresses";

    container mud-acl {
      if-feature mud-acl;
      must "../../../../acl:acl-type = 'ietf-mud:mud-acl'";

      description
        "TODO; improve. MUD-specific matches.";

      leaf manufacturer {
        type inet:host;
        description "authority component of the manufacturer URI";
      }

      leaf same-manufacturer {
        type empty;
```

```
        description "expand to ACEs for each device
                     with the same origin";
      }

      leaf model {
        type string;
        description "specific model (including version) for a
                     given manufacturer";
      }

      leaf local-networks {
        type empty;
        description "this string is used to indicate networks
                     considered local in a given environment.";
      }

      leaf controller {
        type inet:uri;
        description "expands to one or more controllers for a
                     given service that is codified by inet:uri.";
      }

      leaf my-controller {
        type empty;
        description "The network should manage
                     a class of devices related to this MUD-URL that are
                     intended to control this device.";
      }
    }

  }

  augment "/acl:access-lists/acl:acl/" +
    "acl:access-list-entries/acl:ace/" +
    "acl:matches/acl:tcp-acl" {
    description "Adding domain names to matching";
    leaf direction-initiated {
      type direction;
      description "which direction a flow was initiated";
    }
  }
}
<CODE ENDS>
```

7. The Domain Name Extension to the ACL Model

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

The structure of the change is as follows:

```
module: ietf-acldns
  augment /acl:access-lists/acl:acl/acl:access-list-entries \
          /acl:ace/acl:matches/acl:ipv4-acl:
    +--rw src-dnsname?   inet:host
    +--rw dst-dnsname?   inet:host
  augment /acl:access-lists/acl:acl/acl:access-list-entries \
          /acl:ace/acl:matches/acl:ipv6-acl:
    +--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 autjors the ability to control the IP version that the device may utilize.

The following elements are defined.

7.1. source-dnsname

The argument corresponds to a domain name of a source as specified by inet:host. Depending on how the model is used, it may or may not be resolved, as required by the implementation and circumstances.

7.2. destination-dnsname

The argument corresponds to a domain name of a destination as specified by inet:host. Depending on how the model is used, it may or may not be resolved, as required by the implementation and circumstances.

7.3. The ietf-acldns Model

```
<CODE BEGINS>file "ietf-acldns@2016-07-20.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 "2017-07-03" {
  description "Clone across IP ACL types.";
  reference "RFC XXXX: Manufacturer Usage Description
             Specification";
}

revision "2016-07-20" {
  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:access-lists/acl:acl/" +
       "acl:access-list-entries/acl:ace/" +
       "acl:matches/acl:ipv4-acl" {
      description "Adding domain names to matching";
      uses DNS_MATCHES;
    }

    augment "/acl:access-lists/acl:acl/" +
       "acl:access-list-entries/acl:ace/" +
       "acl:matches/acl:ipv6-acl" {
      description "Adding domain names to matching";
      uses DNS_MATCHES;
    }
  }
  <CODE ENDS>
```

8. MUD File Example

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

```
{
  "ietf-mud:metainfo": {
    "lastUpdate": "2017-08-06T18:55:07+02:00",
    "systeminfo": "https://example.com/productinfo/LightingSystem3000",
    "cacheValidity": 168
  },
  "ietf-mud:device": {
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "acl-name": "mud-66805-v6out",
            "acl-type": "ietf-access-control-list:ipv6-acl"
          }
        ]
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
```

```
          {
            "acl-name": "mud-66805-v6in",
            "acl-type": "ietf-access-control-list:ipv6-acl"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:access-lists": {
    "acl": [
      {
        "acl-name": "mud-66805-v6in",
        "acl-type": "ipv6-acl",
        "access-list-entries": {
          "ace": [
            {
              "rule-name": "cl0-in",
              "matches": {
                "ipv6-acl": {
                "ietf-acldns:src-dnsname": "lighting-system.example.com"
                },
                "source-port-range": {
                  "lower-port": 443,
                  "upper-port": 443
                },
                "tcp-acl": {
                  "ietf-mud:direction-initiated": "from-device"
                }
              },
              "actions": {
                "permit": [
                  null
                ]
              }
            }
          ]
        }
      },
      {
        "acl-name": "mud-66805-v6out",
        "acl-type": "ipv6-acl",
        "access-list-entries": {
          "ace": [
            {
              "rule-name": "cl0-out",
              "matches": {
                "ipv6-acl": {
                "ietf-acldns:dst-dnsname": "lighting-system.example.com"
```

```
                },
                "destination-port-range": {
                  "lower-port": 443,
                  "upper-port": 443
                },
                "tcp-acl": {
                  "ietf-mud:direction-initiated": "from-device"
                }
              },
              "actions": {
                "permit": [
                  null
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

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

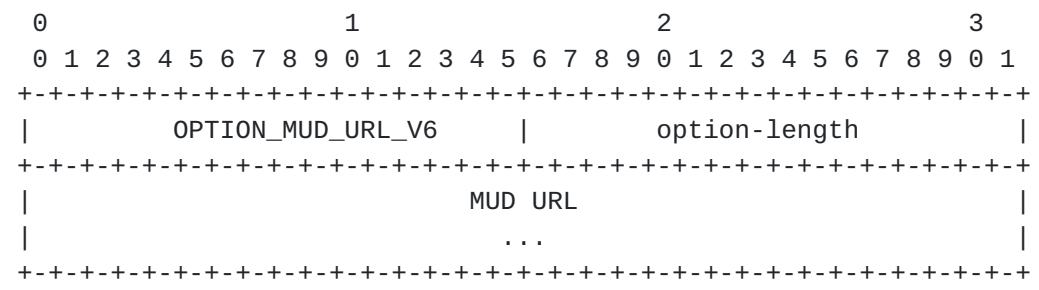
9. The MUD URL DHCP Option

The IPv4 MUD URL client option has the following format:

```
  +------+-----+------------------------------
  | code | len |  MUD URL
  +------+-----+------------------------------
```

Code OPTION_MUD_URL_V4 (161) is assigned by IANA. len is a single octet that indicates the length of the URL in octets. MUD URL is a URL. MUD URLs MUST NOT exceed 255 octets.

The IPv6 MUD URL client option has the following format:



OPTION_MUD_URL_V6 (112; assigned by IANA).

option-length contains the length of the URL in octets.

The intent of this option is to provide both a new device 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 device to the network in a structured way such that the policy can be easily found with existing toolsets.

9.1. Client Behavior

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

Clients SHOULD log or otherwise report improper acknowledgments from servers, but they MUST NOT modify their MUD URL configuration based on a server's response. The server's response is only an acknowledgment that the server has processed the option, and promises no specific network behavior to the client. In particular, it may not be possible for the server to retrieve the file associated with the MUD URL, or the local network administration may not wish to use the usage description. Neither of these situations should be considered in any way exceptional.

9.2. Server Behavior

A DHCP server may ignore these options or take action based on receipt of these options. If a server successfully parses the option and the URL, it MUST return the option with length field set to zero and a corresponding null URL field as an acknowledgment. Even in

this circumstance, no specific network behavior is guaranteed. When a server consumes this option, it will either forward the URL and relevant client information to a network management system (such as the giaddr), or it will retrieve the usage description by resolving the URL.

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

9.3. Relay Requirements

There are no additional requirements for relays.

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

This section defines an X.509 non-critical certificate extension that contains a single Uniform Resource Identifier (URI) 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 URI are defined Section 5 of this document.

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

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

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

The new extension is identified as follows:

```
<CODE BEGINS>

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

  DEFINITIONS IMPLICIT TAGS ::= BEGIN

  -- EXPORTS ALL --

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

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

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

    MUDURLSyntax ::= IA5String

    END

<CODE ENDS>
```

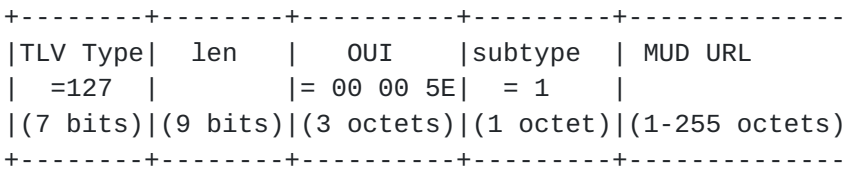
While this extension can appear in either an 802.AR manufacturer certificate (IDevID) or deployment certificate (LDevID), of course it is not guaranteed in either, nor is it guaranteed to be carried over. It is RECOMMENDED that MUD controller implementations maintain a table that maps a device to its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

11. The Manufacturer Usage Description LLDP extension

The IEEE802.1AB Link Layer Discovery Protocol (LLDP) is a one hop vendor-neutral link layer protocols used by end hosts network devices for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network. Its Type-Length-Value (TLV) design allows for 'vendor-specific' extensions to be defined. IANA has a registered IEEE 802 organizationally unique identifier (OUI) defined

as documented in [RFC7042]. The MUD LLDP extension uses a subtype defined in this document to carry the MUD URL.

The LLDP vendor specific frame has the following format:



where:

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

The intent of this extension is to provide both a new device 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 device to the network in a structured way such that the policy can be easily found with existing toolsets.

Hosts, routers, or other network devices 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 devices 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 device and response sent by another device. The other devices do not acknowledge LLDP information received from a device. No specific network behavior is guaranteed. When a device consumes this extension, it may either forward the URL and relevant remote device information to a MUD controller, or it will retrieve the usage description by resolving the URL.

12. Creating and Processing of Signed MUD Files

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

12.1. Creating a MUD file signature

A MUD file MUST be signed using CMS as an opaque binary object. In order to make successful verification more likely, intermediate certificates SHOULD be included. The signature is stored at the same location as the MUD URL but with the suffix of ".p7s". Signatures are transferred using content-type "Application/pkcs7-signature".

For example:

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

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

12.2. Verifying a MUD file signature

Prior to retrieving a MUD file the MUD controller SHOULD retrieve the MUD signature file using the MUD URL with a suffix of ".p7s". For example, if the MUD URL is "https://example.com/.well-known/v1/modela", the MUD signature URL will be "https://example.com/.well-known/v1/modela.p7s".

Upon retrieving a MUD file, a MUD controller MUST validate the signature of the file before continuing with further processing. A MUD controller SHOULD produce an error and it MUST cease all processing of that file if the signature cannot be validated. It is important that MUD controllers have some reason to trust the certificates they are seeing. Therefore, it is RECOMMENDED that new signers be validated either directly by an administrator or by a service that has some reason to believe that the signer is a good actor.

For Example:

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

Note the additional step of verifying the common trust root.

13. Extensibility

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

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

2. At a finer grain, only extensions that would not incur additional risk to the device are permitted. Specifically, augmenting of the metainfo container is permitted with the understanding that such additions may be ignored. In addition, augmentation of the ACL model is permitted so long as it remains safe for a given ACE to be ignored by the MUD Controller or the network elements it configures. Most specifically, one MUST NOT augment behavior of the acl "deny" action. Furthermore, implementations that are not able to parse a component of the ACE array MUST ignore the entire array entry (e.g., not the entire array) and MAY ignore the entire MUD file or otherwise prompt the administrator.

14. Deployment Considerations

Because MUD consists of a number of architectural building blocks, it is possible to assemble different deployment scenarios. One key aspect is where to place policy enforcement. In order to protect the device from other devices within a local deployment, policy can be enforced on the nearest switch or access point. In order to limit unwanted traffic within a network, it may also be advisable to enforce policy as close to the Internet as possible. In some circumstances, policy enforcement may not be available at the closest hop. At that point, the risk of so-called east-west infection is increased to the number of devices that are able to communicate without protection.

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

15. Security Considerations

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

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

It may be possible for a rogue manufacturer to inappropriately exercise the MUD file parser, in order to exploit a vulnerability. There are three recommended approaches to address this threat. The first is to validate the signature of the MUD file. The second is to have a system do a primary scan of the file to ensure that it is both parseable and believable at some level. MUD files will likely be relatively small, to start with. The number of ACEs used by any given device 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 reputations.

Use of a URL necessitates the use of domain names. If a domain name changes ownership, the new owner of that domain may be able to provide MUD files that MUD controllers would consider valid. There are a few approaches that can mitigate this attack. First, MUD controllers SHOULD cache certificates used by the MUD file server. When a new certificate is retrieved for whatever reason, the MUD controller should check to see if ownership of the domain has changed. A fair programmatic approximation of this is when the name servers for the domain have changed. If the actual MUD file has changed, the controller MAY check the WHOIS database to see if registration ownership of a domain has changed. If a change has

occured, or if for some reason it is not possible to determine whether ownership has changed, further review may be warranted. Note, this remediation does not take into account the case of a device that was produced long ago and only recently fielded, or the case where a new MUD controller has been installed.

The release of a MUD URL by a device reveals what the device 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 device, 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 device might otherwise be identified, perhaps through how it behaves when it is connected to the network, whether a device 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 devices may also assist in limiting access to the MUD-URL through the use of mechanisms such as DHCPv6-Shield [RFC7610].

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

16. IANA Considerations

16.1. YANG Module Registrations

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

The ietf-mud module:

- Name: ietf-mud

- XML Namespace: urn:ietf:params:xml:ns:yang:ietf-mud

- Prefix: ief-mud
- Reference: This memo

The ietf-acldns module:

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

16.2. DHCPv4 and DHCPv6 Options

The IANA has allocated option 161 in the Dynamic Host Configuration Protocol (DHCP) and Bootstrap Protocol (BOOTP) Parameters registry for the MUD DHCPv4 option.

IANA is requested to allocated the DHCPv4 and v6 options as specified in Section 9.

16.3. PKIX Extensions

IANA is kindly requested to make the following assignments for:

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

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

The use fo these values is specified in Section 10.

16.4. Well Known URI Suffix

The IANA has allocated the URL suffix of "mud" as follows:

o URI Suffix: "mud" o Specification documents: this document o Related information: n/a

16.5. MIME Media-type Registration for MUD files

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

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

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

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

16.6. LLDP IANA TLV Subtype Registry

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

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

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

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

Reference = < this document >

16.7. 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. 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, and Dan Wing for their valuable advice and reviews. Russ Housley entirely rewrote Section 10 to be a complete module. Adrian Farrel provided the basis for privacy considerations text. Kent Watson provided a thorough review of the archictecture and the YANG model. The remaining errors in this work are entirely the responsibility of the author.

18. References

18.1. Normative References

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-07 (work in progress), July 2017.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Jethanandani, M., Huang, L., Agarwal, S., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-11 (work in progress), June 2017.

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-13 (work in progress), June 2017.

[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, <http://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, <http://www.rfc-editor.org/info/rfc2119>.

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://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, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <http://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, <http://www.rfc-editor.org/info/rfc3986>.

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

[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, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://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, <http://www.rfc-editor.org/info/rfc5905>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[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, <http://www.rfc-editor.org/info/rfc6092>.

[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, <http://www.rfc-editor.org/info/rfc6335>.

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

[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, <http://www.rfc-editor.org/info/rfc7227>.

[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, <http://www.rfc-editor.org/info/rfc7610>.

18.2. Informative References

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

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

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

[RFC1984] IAB and IESG, "IAB and IESG Statement on Cryptographic Technology and the Internet", BCP 200, RFC 1984, DOI 10.17487/RFC1984, August 1996, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc3553>.

[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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7170>.

[RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <http://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, <http://www.rfc-editor.org/info/rfc7488>.

Appendix A. Changes from Earlier Versions

RFC Editor to remove this section prior to publication.

Draft -06 to -07:

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

Draft -06 to -07:

- Update models to match new ACL model
- extract directionality from the ACL, introducing a new device container.

Draft -05 to -06:

- Make clear that this is a component architecture (Polk and Watson)
- Add order of operations (Watson)
- Add extensions leaf-list (Pritikin)
- Remove previous-mud-file (Watson)
- Modify text in last-update (Watson)
- Clarify local networks (Weis, Watson)
- Fix contact info (Watson)
- Terminology clarification (Weis)
- Advice on how to handle LDevIDs (Watson)

- Add deployment considerations (Watson)
- Add some additional text about fingerprinting (Watson)
- Appropriate references to 6087bis (Watson)
- Change systeminfo to a URL to be referenced (Lear)

Draft -04 to -05: * syntax error correction

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

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

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

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

Appendix B. Default MUD elements

What follows is 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 ACEs. To block DNS or NTP one repeats the matching statement but replace "permit" with deny. Because ACEs are processed in the order they are received, the defaults would not be reached. A MUD controller might further decide to optimize to simply not include the defaults when they are overriden.

A complete MUD entry is included below.

```
{
  "ietf-mud:metainfo": {
    "lastUpdate": "2017-08-06T19:12:24+02:00",
    "systeminfo": "https://www.ietf.org/mudcontrollerdefaults",
    "cacheValidity": 168
  },
  "ietf-mud:device": {
    "from-device-policy": {
      "access-lists": {
        "access-list": [
```

```
            {
              "acl-name": "mud-39988-v4out",
              "acl-type": "ietf-access-control-list:ipv4-acl"
            },
            {
              "acl-name": "mud-39988-v6out",
              "acl-type": "ietf-access-control-list:ipv6-acl"
            }
          ]
        }
      },
      "to-device-policy": {
        "access-lists": {
          "access-list": [
            {
              "acl-name": "mud-39988-v4in",
              "acl-type": "ietf-access-control-list:ipv4-acl"
            },
            {
              "acl-name": "mud-39988-v6in",
              "acl-type": "ietf-access-control-list:ipv6-acl"
            }
          ]
        }
      }
    },
    "ietf-access-control-list:access-lists": {
      "acl": [
        {
          "acl-name": "mud-39988-v4in",
          "acl-type": "ipv4-acl",
          "access-list-entries": {
            "ace": [
              {
                "rule-name": "ent0-in",
                "matches": {
                  "ietf-mud:mud-acl"{
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "source-port-range": {
                    "lower-port": 53,
                    "upper-port": 53
                  }
                },
                "actions": {
                  "permit": [
                    null
                  ]
```

```
                }
              },
              {
                "rule-name": "ent1-in",
                "matches": {
                  "ietf-mud:mud-acl"{
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "source-port-range": {
                    "lower-port": 53,
                    "upper-port": 53
                  },
                  "tcp-acl": {
                    "ietf-mud:direction-initiated": "from-device"
                  }
                },
                "actions": {
                  "permit": [
                    null
                  ]
                }
              },
              {
                "rule-name": "ent2-in",
                "matches": {
                  "ietf-mud:mud-acl"{
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "source-port-range": {
                    "lower-port": 123,
                    "upper-port": 123
                  }
                },
                "actions": {
                  "permit": [
                    null
                  ]
                }
              }
            ]
          }
        },
        {
          "acl-name": "mud-39988-v4out",
          "acl-type": "ipv4-acl",
          "access-list-entries": {
            "ace": [
              {
```

```
                "rule-name": "ent0-out",
                "matches": {
                  "ietf-mud:mud-acl"{
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "destination-port-range": {
                    "lower-port": 53,
                    "upper-port": 53
                  }
                },
                "actions": {
                  "permit": [
                    null
                  ]
                }
              },
              {
                "rule-name": "ent1-out",
                "matches": {
                  "ietf-mud:mud-acl"{
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "destination-port-range": {
                    "lower-port": 53,
                    "upper-port": 53
                  },
                  "tcp-acl": {
                    "ietf-mud:direction-initiated": "from-device"
                  }
                },
                "actions": {
                  "permit": [
                    null
                  ]
                }
              },
              {
                "rule-name": "ent2-out",
                "matches": {
                  "ietf-mud:mud-acl"{
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "destination-port-range": {
                    "lower-port": 123,
                    "upper-port": 123
                  }
                },
                "actions": {
```

```
                "permit": [
                  null
                ]
              }
            }
          ]
        }
      },
      {
        "acl-name": "mud-39988-v6in",
        "acl-type": "ipv6-acl",
        "access-list-entries": {
          "ace": [
            {
              "rule-name": "ent0-in",
              "matches": {
                "ietf-mud:mud-acl"{
                  "controller": "urn:ietf:params:mud:dns"
                },
                "source-port-range": {
                  "lower-port": 53,
                  "upper-port": 53
                }
              },
              "actions": {
                "permit": [
                  null
                ]
              }
            },
            {
              "rule-name": "ent1-in",
              "matches": {
                "ietf-mud:mud-acl"{
                  "controller": "urn:ietf:params:mud:dns"
                },
                "source-port-range": {
                  "lower-port": 53,
                  "upper-port": 53
                },
                "tcp-acl": {
                  "ietf-mud:direction-initiated": "from-device"
                }
              },
              "actions": {
                "permit": [
                  null
                ]
```

```
            }
          },
          {
            "rule-name": "ent2-in",
            "matches": {
              "ietf-mud:mud-acl"{
                "controller": "urn:ietf:params:mud:ntp"
              },
              "source-port-range": {
                "lower-port": 123,
                "upper-port": 123
              }
            },
            "actions": {
              "permit": [
                null
              ]
            }
          }
        ]
      }
    },
    {
      "acl-name": "mud-39988-v6out",
      "acl-type": "ipv6-acl",
      "access-list-entries": {
        "ace": [
          {
            "rule-name": "ent0-out",
            "matches": {
              "ietf-mud:mud-acl"{
                "controller": "urn:ietf:params:mud:dns"
              },
              "destination-port-range": {
                "lower-port": 53,
                "upper-port": 53
              }
            },
            "actions": {
              "permit": [
                null
              ]
            }
          },
          {
            "rule-name": "ent1-out",
            "matches": {
              "ietf-mud:mud-acl"{
```

```
                    "controller": "urn:ietf:params:mud:dns"
                  },
                  "destination-port-range": {
                    "lower-port": 53,
                    "upper-port": 53
                  },
                  "tcp-acl": {
                    "ietf-mud:direction-initiated": "from-device"
                  }
                },
                "actions": {
                  "permit": [
                    null
                  ]
                }
              },
              {
                "rule-name": "ent2-out",
                "matches": {
                  "ietf-mud:mud-acl"{
                    "controller": "urn:ietf:params:mud:ntp"
                  },
                  "destination-port-range": {
                    "lower-port": 123,
                    "upper-port": 123
                  }
                },
                "actions": {
                  "permit": [
                    null
                  ]
                }
              }
            ]
          }
        }
      ]
    }
  }
```

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

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

Ralph Droms

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

Dan Romascanu

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