MUD – RFC 8520

Conformance Test Specification

Technical Document
Revision 0.97

UNSW Sydney (Australia) *www.unsw.edu.au*
CNLABS (India, USA, Australia) *www.cnlabs.in*

Table of Contents

Introduction

The exponential growth of vulnerable Internet-of-Things (IoT) devices, like networked cameras, sensors, or actuators, has raised concerns about the cybersecurity risks they pose to individuals, organizations, governments, and the Internet at large. Ensuring the integrity and security of these devices and their networks is crucial and requires immediate attention.

A network-based security approach appears promising and more practical, particularly with the adoption of the Manufacturer Usage Description (MUD) standard ratified by the Internet Engineering Task Force (IETF), which fundamentally aids in reducing network attack surfaces.

This test specification aims to enhance the adoption of this lightweight standard by proposing a systematic approach to verify the conformity of IoT devices with MUD specifications.

Definitions

DUT	Device Under Testing
MUD	Manufacture Usage Description
TLV	Type Length Value
ACE	Access Control Entry
ACL	Access Control List
CA	Certification Authority
URN	Universal Resource Name
URL	Universal Resource Locator
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
DHCP	Dynamic Host Configuration Protocol
TLS	Transport Layer Security
LLDP	Link Layer Discovery Protocol
UTF	Unicode Text Format

Test Organization

This document organizes tests based on related test methodology or goals. This document consists of a series of description blocks; each block describes a single test. The format of the description block is as follows:

Purpose	The **Purpose** is a short statement describing what the test attempts to achieve. It is usually phrased as a simple assertion of the feature or capability to be tested.
References	The **References** section lists cross-references to the specifications and documentation that might be helpful in understanding and evaluating the test and results
Test Setup	The **Test Setup** section describes the configuration of all devices prior to the start of the test. Different parts of the procedure may involve configuration steps that deviate from what is given in the test setup. If a value is not provided for a protocol parameter, then the protocol's default is used for that parameter.
Procedure and Expected Behavior	The **Procedure and Expected Behavior** table contains the step-by-step instructions for carrying out the test. These steps include such things as enabling interfaces, unplugging devices from the network, or sending packets from a test station. The test procedure also cues the tester to make observations of expected behavior, as needed, as not all steps require observation of results. If any behavior is expected for a procedure, it is to be observed prior to continuing to the next step. Failure to observe any behavior prior to continuing constitutes a failed test. Note, that while test numbers continue between test parts, each test part is to be executed independently, and are not cascaded from the previous part.
Possible Problems	The **Possible Problems** section contains a description of known issues with the test procedure, which may affect test results in certain situations.

References

The following documents are referenced in these texts:

[RFC-8520] E. Lear, R. Droms and D. Romascanu, Manufacturer Usage Description Specification, RFC 8520, March 2019.

[RFC-5280] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC5280, May 2008.

[RFC-3986] T. Berners-Lee, R. Fielding and L. Masinter, Uniform Resource Identifier (URI): Generic Syntax, RFC 3986, January 2005.

[RFC-3987] M. Duerst and M. Suignard, Internationalized Resource Identifiers (IRIs), RFC 3987, January 2005.

[RFC-5652] R. Housley, Cryptographic Message Syntax (CMS), RFC 4291, September 2009.

[RFC-2818] E. Rescorla, HTTP Over TLS, RFC 2818, May 2000.

[RFC-2459] R. Housley, W. Ford, W. Polk and D. Solo, Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC 2459, January 1999.

[RFC-8259] T. Bray and Ed., The JavaScript Object Notation (JSON) Data Interchange Format, RFC 8259, December 2017.

[RFC-7951] L. Lhotka, JSON Encoding of Data Modeled with YANG, RFC 7951, August 2016.

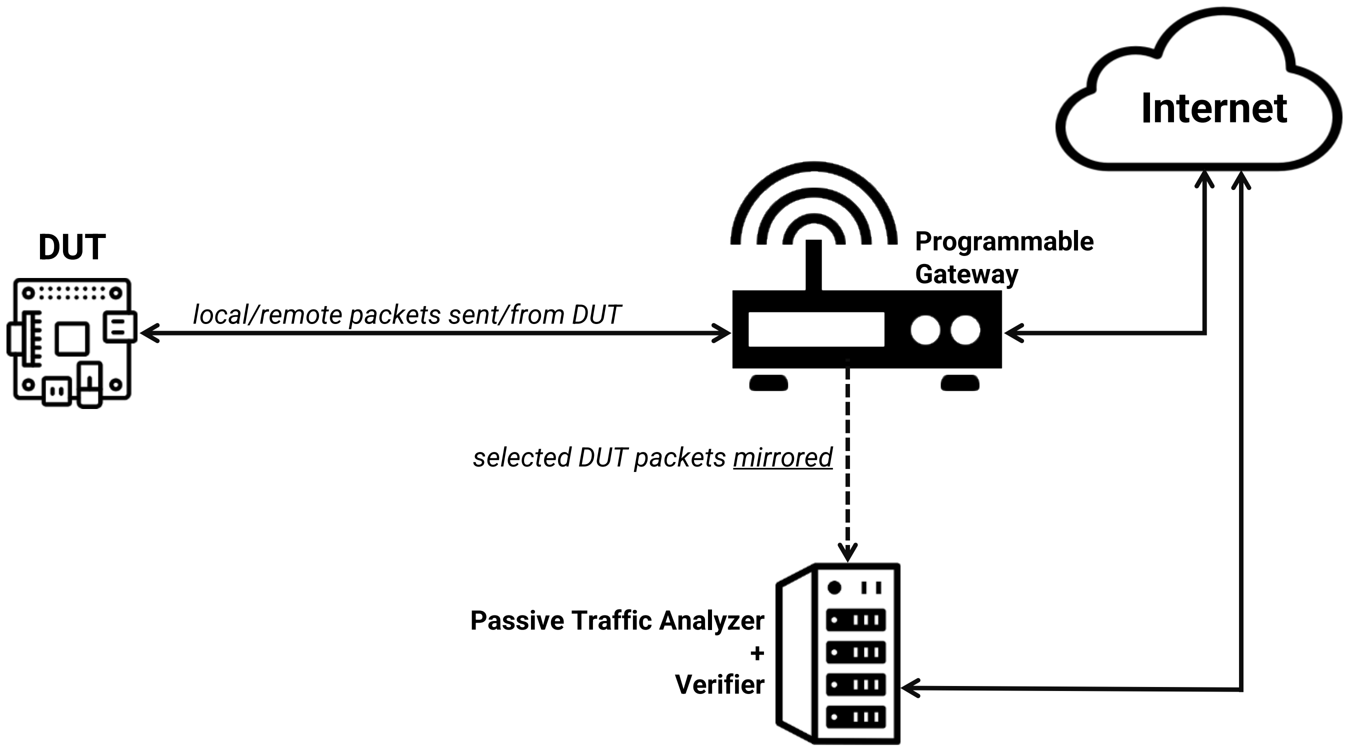
[RFC-3629] F. Yergeau, UTF-8, a transformation format of ISO 10646, RFC 3629, November 2003.

[RFC-8348] A. Bierman, M. Bjorklund, J. Dong and D. Romascanu, A YANG Data Model for Hardware Management, RFC 8348, March 2018.

[RFC-1123] IEFT and R. Braden, Requirements for Internet Hosts -- Application and Support, RFC 1123, October 1989.

[RFC-5905] D. Mills, U. Delaware, J. Martin, Ed., J. Burbank and W. Kasch, Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905, June 2010.

Common Test Topology


Internet
DUT
Programmable Gateway
local/remote packets sent/from DUT
selected DUT packets mirrored
Passive Traffic Analyzer + Verifier

General Requirements

- A server with abilities to (a) capture and analyze packets sent and received by DUT, and (b) verify properties of cloud servers (*e.g.*, storing the MUD file) associated with DUT.
- Network switch/gateway with Wi-Fi and Ethernet interfaces, configured to mirror DUT packets to the Analyzer server.
- DUT has the capability to emit a MUD URL.

Possible Problem Summary

The following test cases have documented possible problems that allow for altered or omitted steps in their procedures. Please see each specific test case listed for more information:

- Test MUD.1.1.1: **Test MUD.1.1.1: Seeking MUD URL String**

Test MUD.1.1.1: Seeking MUD URL String

Purpose: To verify the ability of DUT to emit a MUD URL string.

Functionality Tag: Mandatory

Reference:

- [MUD] – Section 1.5, Section 10, Section 11 and Section 12
- [Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (RFC5280)] – Section 4.2.2.2

Test Setup: Common Test Topology is performed.

Procedure:

Part A: DHCP requests

Step	Action	Expected Behavior
1.	Capture DHCPv4 or DHCPv6 packets sent by DUT and seek a MUD URL for option code 161 in DHCPv4 requests or option code 112 in DHCPv6 requests.	The MUD URL string can be extracted from DHCPv4 or DHCPv6 requests given the corresponding DHCP option code.

Part B: LLDP frames

Step	Action	Expected Behavior
2.	Capture LLDP frames sent by DUT and verify they contain the TLV Type=127, OUI=00 00 5E, subtype=1 and the MUD string as a list of octets	The MUD URL string can be decoded by extracting octet values in the LLDP frames from DUT. The expected format of LLDP packets is shown as follows.

TLV Type = 127 (7 bits)	Len (9 bits)	OUI = 00 00 5E (3 octets)	Subtype = 1 (1 octet)	MUD URL (1-255 octets)

Part C: X.509 Certificates

Step	Action	Expected Behavior
3.	Capture a certificate presented by DUT (*e.g.*, via TLS protocol), look for an X.509 non-critical extension, and verify if it contains a URL.	The MUD URL string can be extracted from the X.509 non-critical extension of the retrieved certificate.

Possible Problems: A manufacturer may choose other means to emit MUD URL. In this case, the usage of protocols and methodology must be declared.

Test MUD.1.1.2: Checking Consistency of Emitted MUD URL Strings

Purpose: Verify that the MUD URL strings emitted by DUT are identical.

Reference:
- [MUD] – Section 10.1

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Compare the extracted MUD URL strings

Step	Action	Expected Behavior
1.	Capture the generated DHCP requests (v4 and/or v6), LLDP frames, and X.509 certificates to check the emitted MUD URLs.	The MUD URL strings extracted from various packet types must be identical

Possible Problems: None.

Test MUD.1.1.3: Validating MUD URL String

Purpose: Verify whether the emitted MUD URL is in a valid format.

Reference:

- [MUD] – Section 1.5, Section 6, Section 10, Section 11, and Section 12
- [Uniform Resource Identifier (URI): Generic Syntax (RFC3986)]
- [Internationalized Resource Identifiers (IRIs) (RFC3987)] – Section 3.1
- [Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (RFC5280)] – Section 7.4

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify the MUD URL string is valid

Step	Action	Expected Behavior
1.	Test whether the MUD URL string starts with "`https`" instead of "`http`".	The MUD URL string must start with "`https`".
2.	Check the length of the MUD URL string is less than a threshold	The length of the MUD URL string must not exceed 255 octets for both DHCP (v4 and/or v6) requests and certificates. Additionally, the length of the MUD URL string must not exceed 253 octets for LLDP frames.
3.	Check whether the URL string has the following structure:	The MUD URL string is expected to adhere to the standard structure.

```
https://domainName/<string>/IRIs
```

where the string only contains legal characters and the domainName should be a valid format with the end such as ".com", the IRIs represents the descriptor of resource, i.e. the name of the MUD file.

Possible Problems: None.

Test MUD.1.1.4: Verifying if a MUD File Can be Retrieved

Purpose: Verify if a MUD file can be retrieved from the specified HTTPS server using the emitted MUD URL.

Functionality Tag: Mandatory

Reference:

- [MUD] – Section 1.6

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Check whether the MUD file can be retrieved

Step	Action	Expected Behavior
1.	Using the GET method, verify whether the MUD file can be retrieved from the HTTPS server specified by the MUD URL.	The GET request for the MUD URL must be successful.

Possible Problems: None.

Test MUD.1.1.5: Verifying Presence and Structure of Digital Signature of MUD File

Purpose: Verify that the retrieved MUD file contains a digital signature and is correctly signed using Cryptographic Message Syntax (CMS).

Reference:

- [MUD] – Section 13.1 and Section 13.2
- [Cryptographic Message Syntax (CMS) (RFC 5652)]

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify the signature method of the MUD file

Step	Action	Expected Behavior
1.	Retrieve the digital signature by obtaining the value of "`mud-signature`" within the MUD file.	The retrieved MUD file should have the node "`mud-signature`" and the signature file can be retrieved from its value.
2.	Check whether the MUD file is signed using DER-encoded CMS, a standard way to digitally sign and encrypt data.	The signature file is a binary object with file type as "`.p7s`" and in DER-encoded format.

Possible Problems: None.

Test MUD.1.1.6: Verifying Content Type and Certificates in Digital Signature of MUD File

Purpose: Verify that content type and certificates within the digital signature of the retrieved MUD file are correct and valid.

Reference:

- [MUD] – Section 11

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify the certificates within the signature

Step	Action	Expected Behavior
1.	Extract the signing certificates in the signature file.	Certificates must exist within the signature file.
2.	Examine the Key Usage Extension pattern within the signing certificate to confirm whether the bit “`digitalSignature`” is set to 0.	The bit “`digitalSignature`” must be set to 0 in the Key Usage Extension pattern.
3.	Check whether the “`id-pe-mudsigner`” extension is present in the X.509 section, and verify if its content matches the information within the subject field of the certificate.	The content of subject in the certificate of the signature must match the content of the “`id-pe-mudsigner`” extension in the retrieved X.509 certificate.

Part B: Verify the content type of the signature

Step	Action	Expected Behavior
4.	Verify if the content-type “`id-ct-mud`” is used as the content type for the signed data.	The content-type is expected to have the object identifier “`id-ct`” equal to 41.

Possible Problems: None.

Test MUD.1.1.7: Verifying Integrity of Digital Signature for MUD File

Purpose: Verify the validity of the digital signature associated with the retrieved MUD file to ensure its integrity and authenticity, thereby confirming that the file has not been tampered with.

Reference:

- [MUD] – Section 13.1, Section 13.2 and Section 11
- [Cryptographic Message Syntax (CMS) (RFC 5652)]

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify the integrity of digital signature

Step	Action	Expected Behavior
1.	Extract the encrypted digital signature and the signer's information, including a public key, a hash function, and a message digest (MD).	
2.	Use the signer's public key and decrypt the digital signature to retrieve the unencrypted hash value (H1).	
3.	Compare the unencrypted hash value with the message digest.	The unencrypted hash value (H1) must match the signer's message digest (MD).

Part B: Verify the integrity of MUD file

Step	Action	Expected Behavior
4.	Calculate the hash value (H0) of the retrieved MUD file using the signer's hash function.	
5.	Compare the calculated hash value (H0) with the signer's message digest (MD).	The hash value of the MUD file (H0) must match the signer's message digest (MD).

Possible Problems: None.

Test MUD.1.1.8: Verifying Certificate Validity of MUD File Signature

Purpose: Verify the validity of the certificate from the retrieved MUD signature.

Functionality Tag: Mandatory

Reference:
- [MUD] – Section 1.6, Section 13.1
- [HTTP Over TLS (RFC 2818)] – Section 3.1
- [Internet X.509 Public Key Infrastructure Certificate and CRL Profile (RFC 2459)]

Test Setup: Common Test Topology is performed.

X.509 Certificate	
Patterns to be verified	Expected Results
The digital signature applied by the Certificate Authority (CA) using its own private key on the certificate needs to be verified using the CA's corresponding public key	When the certificate data, excluding the signature, is encrypted using the algorithm specified in the certificate, the result should match the value obtained by decrypting the digital signature with the CA's public key
The validity period of the X.509 certificate does not expire	The current system time should not before the start date or after the end date shown on the certificate
The revocation information of the certificate needs to be checked to ensure the certificate is not revoked	The certificate is still active and not revoked
The validity of each certificate along the chain of trust associated with the Certificate Authority (CA) responsible for issuing the certificate need to be checked	The verification process is successfully completed when it reaches the root of the certificate chain without encountering any self-signed certificates, and all certificates in the chain are confirmed as valid

Procedure:

Part A: Verify the integrity of digital signature

Step	Action	Expected Behavior
1.	Obtain the intermediate certificate from the retrieved signature.	There should include an intermediate X.509 certificate from the retrieved signature.

2.	Verify the obtained certificate is valid using the matching rules listed above.	The obtained X.509 certificate must be valid.

Possible Problems: None.

Test MUD.1.2.1: Verifying MUD File Format

Purpose: Verify whether the content of the retrieved MUD file is in JSON format.

Functionality Tag: Mandatory

Reference:

- [MUD] – Section 1.6, Section 2 and Section 17
- [The JavaScript Object Notation (JSON) Data Interchange Format (RFC8259)]
- [JSON Encoding of Data Modeled with YANG (RFC7951)]

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Checking the file format used by the MUD file

Step	Action	Expected Behavior
1.	Examine the syntax and structure of the retrieved MUD file.	The content of the retrieved MUD file is expected to be in JSON format.

Possible Problems: None.

Test MUD.1.2.2: Verifying Encoding Method of MUD File

Purpose: Verify that the retrieved MUD file is encoded in UTF-8.

Reference:

- [MUD] – Section 17.5
- [UTF-8, a transformation format of ISO 10646 (RFC3629)]

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Checking the encoding method used by the MUD file

Step	Action	Expected Behavior
1.	Examine the encoding method applied to the retrieved MUD file.	The retrieved MUD file must be encoded in UTF-8.

Possible Problems: None.

Test MUD.1.2.3: Verifying Usage of Nodes in MUD File

Purpose: Verify that the retrieved MUD file includes only certain specified nodes, except optional nodes defined by extensions.

Functionality Tag: Mandatory

Reference:

- [MUD] – Section 2, Section 2.1 and Section 3.9

Test Setup: Common Test Topology is performed.

MUD file structure	Specified Node Names
"mud" container	`mud-version, mud-url, last-update, mud-signature, cache-validity, is-supported, systeminfo, mfg-name, model-name, firmware-rev, software-rev, documentation, extensions, from-device-policy, to-device-policy`
Augmentation to "acl"	`manufacturer, same-manufacturer, model, local-networks, controller, my-controller, direction-initiated`

Procedure:

Part A: Check whether the MUD file only includes certain specified nodes

Step	Action	Expected Behavior
1.	Retrieve all node names from the retrieved MUD file.	
2.	Compare all node names with the table above.	The nodes utilized in the retrieved MUD file must be restricted to those listed in the table above.

Possible Problems: None.

Test MUD.1.2.4: Verifying Extension Declaration in MUD File

Purpose: Verify whether extensions used in the retrieved MUD file are explicitly mentioned in the "`extension`" list. Additionally, ensure that undeclared MUD extensions are not utilized in the retrieved MUD file.

Reference:

- [MUD] – Section 3.9, Section 14, Section 17.8

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify that all the extensions used in the MUD file are explicitly named in the "extension" list

Step	Action	Expected Behavior
1.	Compare each extension with those listed in the "`extension`" list.	The extensions used must match those listed in the "`extension`" list.

Part B: Verify the usage of undeclared MUD extensions in the MUD file

Step	Action	Expected Behavior
2.	Obtain the nodes of the MUD file.	
3.	Verify the existence of undeclared MUD extensions within the nodes of the retrieved MUD file.	The retrieved MUD file must not use undeclared MUD extensions in its nodes.

Possible Problems: None.

Test MUD.1.2.5: Verifying Extensions are Standardized by IETF

Purpose: Verify if all extensions declared and used in the MUD file have undergone the IETF standardization process and are registered with the IANA.

Reference:

- [MUD] – Section 14

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Check whether all the extensions used in the MUD file are standardized by IETF

Step	Action	Expected Behavior
1.	Obtain all the extensions declared and used.	
2.	Verify if the structure of each extension conforms to the format specified as follows:	All extensions used in the retrieved MUD file must be referenced to a published RFC document.

```
Extension name: UTF-8-encoded string, not to exceed 40
characters.
Each Extension must refer to a certain RFC documentation
```

Possible Problems: None.

Test MUD.1.3.1: Verifying Direction of Individual Policies in MUD File

Purpose: Verify that the MUD file explicitly delineates communication directions in each policy, indicating what is to be permitted or denied in either direction of communication, i.e., to-device and from-device. Additionally, verify the accurate specification of policies for their respective communication directions.

Functionality Tag: Mandatory

Reference:

- [MUD] – Section 3.3, Section 7

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Validate directional policies

Step	Action	Expected Behavior
1.	Verify the retrieved MUD file explicitly declares the "`from-device-policy`" and "`to-device-policy`" nodes.	The "`from-device-policy`" node and "`to-device-policy`" node must be declared.

Part B: Validate the correct policy positions

Step	Action	Expected Behavior
2.	Examine individual ACLs within the "`from-device-policy`" and "`to-device-policy`" to ensure that the direction of each ACL aligns with the corresponding policy.	The ACLs within the "`from-device-policy`" are expected to specify communications originated from the DUT and the ACLs within the "`to-device-policy`" are expected to specify communications destined to the DUT.

Possible Problems: None.

Test MUD.1.3.2: Verifying Node cache-validity in MUD File

Purpose: Verify that the "`cache-validity`" value in the retrieved MUD file is within the valid range.

Reference:

- [MUD] – Section 3.5, Section 7

Test Setup: Common Test Topology is performed.

cache-validity

`Type: uint8`
`Value: must not exceed 168, recommended to be no less than 24, should be no shorter than any period determined through HTTP caching directives.`

HTTP caching directives	
Headers in GET response	How to derive the value of HTTP caching-directives
Expires	= the value of "`Expires`" header – the system time that the response is received
Cache-Control	= the value of "`maxage`" attribute – the value of "`age`" attribute
	= if the attribute "`s-maxage`" exists, it takes the highest priority. In this case, the values specified by the max-age or Expires are ignored, and the value of "`s-maxage`" is used

Procedure:

Part A: Verify the value of the node "`cache-validity`" specified in the MUD file

Step	Action	Expected Behavior
1.	Extract the value of the "`cache-validity`" node.	
2.	Obtain the value the HTTP caching directives using the table above.	The value of the HTTP caching directives can be retrieved.
3.	Compare the value of cache-validity with the valid range [24, 168] against the values obtained from the HTTP caching directives.	The value of the cache-validity must be equal to or smaller than 168 and is recommended to be greater than 24. Additionally, the value of the cache-validity should be greater than the value of the retrieved HTTP caching directives.

Possible Problems: None.

Test MUD.1.3.3: Verifying Node systeminfo in MUD File

Purpose: Verify that the values of the "`systeminfo`" node in the retrieved MUD file, describing the DUT, are appropriately set.

Functionality Tag: Mandatory

Reference:

- [MUD] – Section 3.7, Section 7
- [UTF-8, a transformation format of ISO 10646 (RFC3629)]

Test Setup: Common Test Topology is performed.

systeminfo

```
Type: UTF-8
Length: should not exceed 60 characters worth of display space.
```

Procedure:

Part A: Verify the type and length of the "`systeminfo`" values

Step	Action	Expected Behavior
1.	Obtain the values of the "`systeminfo`" node.	
2.	Check the Type (encoding method).	The encoding method used is UTF-8.
3.	Check the Length of description.	The length of the "`systeminfo`" value should not exceed 60 characters, excluding white spaces.

Possible Problems: None.

Test MUD.1.3.4: Verifying Optional Fields in MUD File

Purpose: Verify that the “`firmware-rev`” and “`software-rev`” fields are not included in the MUD file if the device can be upgraded, but the MUD URL cannot be upgraded.

Reference:

- [MUD] – Section 3.6, Section 3.8
- [A YANG Data Model for Hardware Management (RFC8348)]

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Check the optional fields of the MUD file

Step	Action	Expected Behavior
1.	Obtain the Boolean value of the node “`is-supported`” in the MUD file.	
2.	Check whether the optional fields “`firmware-rev`” and “`software-rev`” are present in the MUD file.	If the Boolean value of “`is-support`” is false, the optional fields “`firmware-rev`” and “`software-rev`” must not exist in the MUD file.

Possible Problems: None.

Test MUD.1.4.1: Verifying Format of Local Addresses in MUD File

Purpose: To ensure that local addresses in the “`local-network`” attribute in the ACEs from the retrieved MUD file conform to the acceptable formats, including IPv4/v6 addresses with prefixes and masks (e.g., 255.255.255.255/32).

Reference:

- [MUD] – Section 7

Test Setup: Common Test Topology is performed.

Local Addresses	
Valid IPv4 Addresses format	`Prefix/mask: xxx.xxx.xxx.xxx/yy` where `xxx` is a decimal number within the range [0,255], separated by the dots, followed by a slash and yy, which can be either the CIDR (Classless Inter-Domain Routing) notation that is a decimal number within the range [0, 32] or the Traditional Subnet Mask Format that is the same format as the prefix pattern.
Valid IPv6 Addresses format	`Prefix/mask: xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx/yyy` where `xxxx` is hexadecimal number ranging from 0 to FFFF, separated by the colons, followed by a slash the and yyy, which is the CIDR (Classless Inter-Domain Routing) notation that is a decimal number within the range [0, 128].

Procedure:

Part A: Verify local address formats

Step	Action	Expected Behavior
1.	Retrieve the ACEs from the MUD file that reference “`local-network`”.	
2.	Examine the IP addresses within those ACEs to confirm they include valid formats with prefixes and masks.	Local addresses in the “`local-network`” attribute in the ACEs may be specified with valid prefixes and masks, as indicated in the table above.

Possible Problems: None.

Test MUD.1.4.2: Verifying URLs of Controllers in MUD File

Purpose: Examine the controller URLs to ensure they are valid or in the form of URNs.

Reference:

- [MUD] – Section 4.6, Section 17.7
- [Requirements for Internet Hosts -- Application and Support (RFC1123)]
- [Network Time Protocol Version 4: Protocol and Algorithms Specification (RFC5905)]

Test Setup: Common Test Topology is performed.

controller	
MUD Well-Known Universal Resource Name (URNs)	"`urn:ietf:params:mud:dns`", "`urn:ietf:params:mud:ntp`"
Valid URL form	"`http[s]://domainName/<string>`", where the domainName is in a valid format with endings, such as "`.com`"

Procedure:

Part A: Check the URL of the node "`controller`"

Step	Action	Expected Behavior
1.	Obtain the URL of the "`controller`" nodes in the ACLs.	
2.	Compare these URL values with the MUD well-known URNs.	The controller URL may take the form of MUD well-known URNs.
3.	Verify the validity of values if they are in the form of URLs.	The controller URL may take the form of a valid URL.

Possible Problems: None.

Test MUD.1.5.1: Verifying Usage of ACL Leaf Nodes in MUD File

Purpose: Verify that the ACL leaf nodes include the required attributes, specifically: “`name`” of the ACL, “`type`”, “`name`” of the ACEs, and TCP and UDP source and destination port information from the retrieved MUD file.

Reference:

- [MUD] – Section 2

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify inclusion of ACL attributes in the MUD File

Step	**Action**	**Expected Behavior**
1.	Examine all ACL entries within the retrieved MUD file to ensure they include the following attributes: “`name`” of the ACL, “`type`”, “`name`” of the ACEs, and source and destination port information for ACEs based on TCP or UDP.	Each ACL should contain the specified attributes: “`name`” of the ACL, “`type`”, “`name`” of the ACEs, and source and destination port information for ACEs based on TCP/UDP.

Possible Problems: None.

Test MUD.1.5.2: Verifying ACL Policies in MUD File

Purpose: Verify that each ACE policy in the MUD file explicitly defines rules for either inbound or outbound traffic relative to the DUT.

Reference:

- [MUD] – Section 7

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify Inbound and/or Outbound Traffic Policies in ACLs

Step	Action	Expected Behavior
1.	Examine each ACE in the retrieved MUD file to verify that there are policies governing either inbound (to-device) or outbound (from-device) traffic in relation to the DUT by inspecting the `“name”` attribute of each ACL.	Each ACL should clearly specify policies for either inbound or outbound traffic with respect to the DUT in their `“name”` attribute which contains either “from” or “to”.

Possible Problems: None.

Test MUD.1.5.3: Verifying Usage of Domain Names in ACEs

Purpose: Verify that ACEs with "`ietf-acldns:src-dnsname`" and "`ietf-acldns:dst-dnsname`" for endpoints are configured by valid domain names.

Reference:

- [MUD] – Section 2, Section 8

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify dnsname extensions for endpoints in ACEs are configured by domain names

Step	Action	Expected Behavior
1.	Retrieve those ACEs in the MUD file that specify the endpoints using DNS attributes: "`ietf-acldns:src-dnsname`" or "`ietf-acldns:dst-dnsname`"	
2.	Confirm the values of these attributes are set by valid domain names, not IP addresses.	Each ACE that utilizes the DNS extensions should specify the endpoint address using a domain name, not an IP address.

Possible Problems: None.

Test MUD.1.5.4: Verifying Consistency of Policies in ACLs

Purpose: Verify that attributes of the policy specified by each ACE is consistent with their intent in the MUD file.

Reference:

- [MUD] – Section 8.1 and Section 8.2

Test Setup: Common Test Topology is performed.

Device policies	Allowed attributes	Disallowed attributes
from-device	`dst-dnsname`, `destination-ipv4-network`, `destination-ipv6-network`	`src-dnsname`, `source-ipv4-network`, `source-ipv6-network`
to-device	`src-dnsname`, `source-ipv4-network`, `source-ipv6-network`	`dst-dnsname`, `destination-ipv4-network`, `destination-ipv6-network`

Procedure:

Part A: Verify the policy criteria used for "from-device" policies.

Step	Action	Expected Behavior
1.	Examine all ACEs within the ACL container in the MUD file where the policy is defined as "`from-device`"	
2.	Verify each ACE using "`from-device`" employs only allowed attributes: "`dst-dnsname`", "`destination-ipv4-network`" or "`destination-ipv6-network`" to signify the destination endpoint and does not utilize any of the disallowed attributes listed in table above.	All "`from-device`" ACEs MUST utilize allowed attributes: "`dst-dnsname`", "`destination-ipv4-network`" or "`destination-ipv6-network`" for endpoints, and there is no instance of disallowed attributes in this context.

Part B: Verify the policy criteria used for "to-device" policies.

Step	Action	Expected Behavior
3.	Examine all ACEs within the ACL container in the MUD file where the policy is defined as "`to-device`"	
4.	Verify each ACE using "`to-device`" employs only allowed attributes: "`src-dnsname`", "`source-ipv4-network`" or	All "`to-device`" ACEs MUST utilize allowed attributes: "`src-dnsname`", "`source-ipv4-network`" or "`source-ipv6-network`" for endpoints, and there

	“`source-ipv6-network`” to signify the source endpoint and does not utilize any of the disallowed attributes listed in table above.	is no instance of disallowed attributes in this context.

Possible Problems: None.

Test MUD.1.5.5: Verifying Usage of "`direction-initiated`" for ACEs in MUD File

Purpose: Verify that the "`direction-initiated`" attribute is exclusively applied to TCP-based ACEs in the retrieved MUD file.

Reference:

- [MUD] – Section 4.8, Section 9

Test Setup: Common Test Setup is performed.

Procedure:

Part A: Verify "`direction-initiated`" is applied TCP connections.

Step	Action	Expected Behavior
1.	Ensure that the attribute "`direction-initiated`" is only associated with TCP-based ACEs in the retrieved MUD file.	The "`direction-initiate`" must only apply to TCP communications (*i.e.*, ACEs with `protocol=6`).

Possible Problems: None.

Test MUD.1.5.6: Verifying Actions of ACEs in MUD File

Purpose: Verify that each ACE in the retrieved MUD file specifies only “`accept`” or “`drop`” as actions without additional information.

Reference:

- [MUD] – Section 2

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Verify the “`actions`” attribute of ACEs.

Step	Action	Expected Behavior
1.	Extract all ACEs from the MUD file.	
2.	Verify that the “actions” attribute of each ACE in the MUD file specifies only “`accept`” or “`drop`”.	Each ACE in the retrieved MUD file should only specify “`accept`” or “`drop`” for the “`actions`” attribute.

Possible Problems: None.

Test MUD.1.5.7: Verifying Count of ACEs in MUD File

Purpose: Verify that the number of ACEs in the retrieved MUD file is “relatively” small.

Reference:

- [MUD] – Section 16

Test Setup: Common Test Topology is performed.

Procedure:

Part A: Check the number of ACEs in the MUD file

Step	Action	Expected Behavior
1.	Obtain all the ACEs in the retrieved MUD file.	
2.	Count the number of ACEs in the retrieved MUD file.	The number of ACEs in the retrieved MUD file should be relatively small (*e.g.*, less than 50).

Possible Problems: None.

Modification Record

Version 0.97 ***June 29, 2024***