IOTOPS Working Group
Internet-Draft
Updates: 8520 (if approved)
Intended status: Standards Track
Expires: 4 November 2026

M. Richardson
Sandelman Software Works
W. Pan
Huawei Technologies
E. Lear
Cisco Systems

3 May 2026

Authorized update to MUD URLs
draft-ietf-iotops-mud-acceptable-urls-02

Abstract

This document provides a way for an RFC8520 Manufacturer Usage Description (MUD) definitions to declare what are acceptable replacement MUD URLs for a device.

RFCEDITOR-please-remove: this document is being worked on at: https://github.com/mcr/iot-mud-acceptable-urls

Status of This Memo

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

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

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

This Internet-Draft will expire on 4 November 2026.

Copyright Notice

Table of Contents

1. Introduction

[RFC8520] provides a standardized way to describe how a specific purpose device makes use of Internet resources and associated suggested network behavior. The behaviors are described in a MUD file hosted in its manufacturer's server. The device provides a MUD URL to the MUD controller, which can locate this MUD file and determine the required network authorization of the device.

In some cases, e.g., the firmware update, the network behaviors of the device may change, and the description in the original MUD file will no longer apply. To solve this problem, there are two common ways which the manufacturer can use.

One is to change what is in the MUD file, i.e., update the MUD file in place, whenever the behavior of the firmware changes. Section 3 discusses three scenarios for updating the MUD file and the corresponding potential issues.

The other is to change which MUD file is processed by changing the MUD URL. Section 4 describes the common sources of MUD URLs and the problems and threats faced by each type of source when updating the MUD URL. This document proposes an enhanced mechanism of how to securely update the MUD URL in Section 5.

There are also some assumptions and prerequisites in this document.

While MUD files may include signatures, [RFC8520] does not mandate checking them, and there is not a clear way to connect the entity which signed the MUD file to the device itself. This document limits itself to situations in which the MUD file is signed, and that the MUD controller has been configured to always check the signatures, rejecting files whose signatures do not validate.

[RFC8520] does not specify how MUD controllers establish their trust in the manufacturers' signing key: there are many possible solutions from manual configuration of trust anchors, some kind of automatic configuration during onboarding, or a Trust on First Use (TOFU) mechanism that accepts the signer on first use. How this initial trust is established is not important for this document, it is sufficient that some satisfactory initial trust is established.

2. Terminology

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

3. Risk analysis of updating the MUD files in place

This section explains three scenarios where updating the MUD file in place could cause security issues for the devices involved. This section explains why changing the MUD URL to point to a new file is important.

3.1. Adding capabilities

For situations where new capabilities are added to the firmware, the MUD file will detail the new access that the new firmware requires. This may involve new incoming or outgoing connections that should be authorized. Devices that have been upgraded to the new firmware will make use of the new features. Devices that have not been upgraded to the new firmware may have new connections that are authorized, but which the device does not use (outgoing connections), or which the device is not prepared to respond to (new incoming connections).

It is possible that older versions of the firmware have vulnerabilities that were not easily exploitable due to the MUD file preventing particular kinds of access. For example, an older firmware could have no credentials required (or default credentials) access via telnet on port 23 or HTTP on port 80. The MUD file protected the device such that it could either not be accessed at all, or access was restricted to connections from a controller only.

Useful and needed upgrades to the firmware could add credentials to that service, allowing it to be opened up for more general access. The new MUD file would provide for such access, but when combined with the weak security of the old firmware, it results in a compromised device.

While there is an argument that old firmware was insecure and should be replaced, it is often the case that the upgrade process involves downtime, or can introduce risks due to needed evaluations not having been completed yet.

3.2. Removing capabilities

For situations where existing capabilities prove to be a problem and are to be turned off or removed in subsequent versions of the firmware, the MUD file will be updated to disallow connections that previously were allowed.

In this case, the new MUD file will forbid some connections, which the old firmware still expects to do. As explained in the previous section, upgrades may not always occur immediately upon releasing the new firmware.

In this case, the old device will be performing unwanted connections, and the MUD controller will be alerting the network owner that the device is misbehaving rather than not being upgraded. This causes a false-positive situation (see [boycrieswolf]), leading to real security issues being ignored. This is a serious issue as documented also in [boywolfinfosec], and [falsemalware].

3.3. Significant changes to protocols

[I-D.ietf-opsawg-mud-tls] suggests MUD definitions to allow examination of TLS protocol details. Such a profile may be very specific to the TLS library which is shipped in a device. Changes to the library (including bug fixes) may cause significant changes to the profile, requiring changes to the profile described in the MUD file. Such changes are likely neither forward nor backward compatible with other versions, and in place updates to MUD files are therefore not advised.

3.4. Motivation for updating MUD URLs

While many small tweaks to a MUD file can be done in place, the situation described above, particularly when it comes to removing capabilities will suggest that changes to the MUD URL are in order. A strategy for doing this securely is needed, and the rest of this document provides a mechanism to do this securely.

4. Updating the MUD URLs

MUD URLs can come from a number of sources:

* IDevID Extensions
* DHCP option
* LLDP TLV
* [RFC9238] standardizes scanning MUD URLs from QRcodes.

The IDevID mechanism provides a URL that is asserted cryptographically by a manufacturer. However, it is difficult for manufacturers to update the IDevID of a device which is already in a box.

The DHCP and LLDP mechanisms are not signed, but are asserted by the device. A firmware update may update what MUD URL is emitted. Sufficiently well targeted malware would also be able to change the MUD URL that is emitted.

The QRcode mechanism is usually done via paper/stickers, and is typically not under the control of the device itself at all. However, being applied by a human and not easily changed, a MUD URL obtained in this fashion is likely as trustworthy as the rest of the vendors packaging. (It may not, due to mixups in labeling represent the correct device, but this is a human coordination issue, and is out of scope for this document.)

The manufacturer can use all the four mechanisms above when manufacturing the device. But when considering updating the firmware, it seems like only the DHCP and LLDP mechanisms are sufficiently easy to send the new MUD URL. Because of that sensitivity, they may also be easily changed by malware!

There are mitigating mechanisms which may be enough to deal with this problem when MUD files are signed by the manufacturer.

[RFC8520], Section 13.2 explains how to verify MUD File Signatures. That document does not define a way for a MUD controller to determine who should sign the MUD file for a particular device.

[RFC8520] leaves this for a local policy. This document establishes one such local policy. There are a number of other processes that could be used, it is expected that many such industrial vertical will work out supply chain arrangements or other heuristics to supply appropriate anchors.

4.1. Leveraging the manufacturer signature

The first time a signature of the MUD file related to a particular device-type is verified by the MUD controller, the identity of the signing authority is recorded. That it, the signing authority is pinned. This policy means that subsequent MUD files must be signed by the same entity in order to be accepted.

The trust and acceptance of the first signer may come from many sources. The first signature could be from a manually configured trust anchor in the MUD controller. The first signature could be Trust on First Use (TOFU), with the URL coming from a trusted IDevID certificate.

Based upon this process, an update to the MUD URL would be valid if the new MUD file was signed by the same entity that signed the previous entry. This mechanism permits a replacement URL to be any URL that the same manufacturer can provide.

4.2. Concerns about same-signer mechanism

There is still a potential threat: a manufacturer which has many products may have a MUD definition for another product that has the privileges that the malware desires.

The malware could simply change the MUD URL expressed by DHCP or LLDP to that of another product, and it will be accepted by the MUD controller as valid, since the MUD file is signed by the same manufacturer. (e.g., The malware in a BrandName refriderator claims to be a BrandName Washing Machine, and therefore gets the priviledges of the other device)

This works as long as manufacturers use a single key to sign all products. Some manufacturers could sign each product with a different key. Such manufacturers would probably then collect all the signing keys into a certificate infrastructure (PKI), with a single manufacturer CA key.

In this case, the question then becomes whether the MUD controller should pin the End-Entity (EE) certificate, or the CA certificate.

Pinning the End-Entity (EE) certificate defends against malware that changes the product type, but prevents the manufacturer from being able to cycle the validity of the End-Entity certificate for cryptographic hygiene reasons.

Pinning the CA certificate allows the EE certificate to change, but may not defend against product type changes.

It is possible to invent policy mechanisms that would link the EE certificate to a value that is in the MUD file. This could be a policy OID, or could involve some content in a subjectAltName. Future work could go in that direction. This document proposes a simpler solution.

5. Proposed mechanism for updating MUD URLs

The document proposes to limit what MUD URLs are considered valid from the device, limiting new MUD URLs to be variations of the initial (presumed to be secure) URL.

The first MUD file which is defined for a device can come from an IDevID (which is considered more secure), or via Trust on First Use with DHCP or LLDP or other mechanisms. This first, initially trusted, MUD file will be called the "root" MUD file.

A MUD file contains a self-referential MUD-URL attribute that points to the MUD file itself located on the vendor’s website. While the IDevID, DHCP and LLDP mechanisms only transmit a URL, there are some newer, not yet standardized proposals that would fetch an entire MUD file from the device, such as [I-D.jimenez-t2trg-mud-coap].

The MUD-URL MUST always be an Absolute URI: see [RFC3986] section 4.3.

The URL found in the MUD-URL attribute is to be called the canonical MUD URL for the device.

The MUD-SIGNATURE attribute in the MUD file SHOULD be a relative URI (see [RFC3986] section 4.2) with the (hierarchical) base URI for this reference being the MUD-URL attribute.

When pinning the signature, the MUD manager SHOULD use the SubjectKeyIdentifier (SKI) [RFC5280], Section 4.2.1.2 of the Certificate Authority (CA) when pinning the certificate authority. With this, the chain of Subject Names and/or SubjectAltNames leading to the (end entity) signing certificate needs to be recorded. The MUD manager may need additional trust anchors (including previously pinned ones) in order to verify that CA certificate. This process allows for the manufacturer to generate new end-entity signing certificates, as the certificates for this key expire.

5.1. Small Changes to the MUD URL

Subsequent MUD files are considered valid if:

* they have the same initial Base-URI as the MUD-URL, but may have a different final part

* they are signed by an equivalent End Entity (same trusted CA and same Subject Name) as the "root" MUD file.

Section 5.2 of [RFC3986] details many cases for calculating the Base-URI.

Section 3.3 of [RFC3986] explains how the different parts of the URL are described. As explained in that section, a _path_ component consists of a series of _segment_ seperated by slash ("/") characters. The new URL is considered acceptable if it contains the same series of segments in its path, excepting that the last segment may be different.

For a simple example, if the canonical MUD-URL is http://example.com/hello/there/file.json then any URL that starts with http://example.com/hello/there/ would be acceptable, such as http://example.com/hello/there/revision2.json

One problem with these small changes is that malware could still express a MUD file that was previously valid, but which should no longer considered accurate. This is a rollback attack. This might

result in the malware being able to reach destinations that turned out to be a mistake; a security fault. In order to combat this, MUD managers SHOULD keep track of the list of MUD-URLs that they have successfully retrieved, and if a device ever suggests a URL that was previously used, then the MUD manager should suspect that is a rollback attack. MUD managers are not typically resource constrained, and while the list of URLs could grow without bound, it is unlikely to be a burden. A site with thousands of similar devices could keep a common list of URLs.

5.2. Big Changes to the MUD URL

Once a new MUD file is accepted, either by reloading an existing file from the same URL, or via the Small Changes mechanism described above, then the MUD-URL attribute in this file becomes the new canonical MUD file. The contained MUD-URL attribute in the file need not be related in any way to the existing MUD-URL.

As a result, any subsequent updates MUST be relative to the new MUD-URL in this file.

This rule enables the location of the MUD file to change over time based upon the needs of the organization.

5.3. Merger, Acquisitions and Key Changes

The above process allows for a manufacturer to rework its file structure. They can change web server host names, so long as they retain the old structure long enough for all devices to upgrade at least once.

The process also allows a manufacturer to change the EE certificate and Certification Authority used for signing.

5.3.1. Changing file structure

A manufacturer has been hosting a MUD file at https://example.com/household/products/mudfiles/toaster.json and wishes to move it to https://example.com/mudfiles/toasters/model1945/mud.json

The manufacturer creates a new MUD file at the new location.

Then the manufacturer changes the MUD-URL contained with the files at the old location to have a value of https://example.com/mudfiles/toasters/model1945/mud.json Note that in order for MUD controllers to reload the old file, it MUST have been served with an appropriate ETag, and appropriate Expires or Cache

Control headers [RFC9111], Section 5.3. If control over caching is not possible for the manufacturer, then they need to do this in two steps, with the first step creating a new MUD file at an acceptable location (in the above example, perhaps: https://example.com/household/products/mudfiles/toaster0.json). The device then will have to do two firmware updates: one to switch to the intermediate URL, and a second one to switch to the desired final URL.

The manufacturer must continue to serve the files from the old location for some time, or to return an HTTP 301 (Moved Permanently) redirecting to the new location.

5.3.2. Changing hosting URLs

A manufacturer has been hosting a MUD file at https://example.com/household/products/mudfiles/toaster.json and wishes to move it to https://mud.example/hosthold/products/mudfiles/toaster.json

The scenario is much the same as for Section 5.3.1, and can be handled in the same fashion. This situation is likely to occur when one company acquires another.

Note, however, that a 301 Redirect that changed the hostname SHOULD NOT be accepted by MUD controllers.

5.3.3. Changing Signing Authority

A manufacturer has been signing MUD files using an EE Certificate with subjectAltName foo.example, issued by an internal Certification Authority BAZ.

The manufacturer wishes to begin signing with an EE Certificate with subjectAltname foo.example, but now signed by a public CA (call it: Fluffy).

The manufacturer first creates a new MUD file with a new detached signature file. Within this signature file, the manufacturer places a certificate chain: Internal-CA BAZ->Fluffy, and then the Fluffy Certificate, and then the foo.example certificate issued from Fluffy.

This supports changing certification authorities, but it does not support changing the Subject Name of the signing entity.

6. Polling for changes in MUD files

The MUD file update mechanisms described in Section 3 requires that the MUD controller poll for updates. The MUD controller will receive no signal about a change from the device because the URL will not have changed.

The manufacturer SHOULD serve MUD files from a source for which ETag Section 2.3 of [RFC7232] may be generated. Static files on disk satisfy this requirement. MUD files generated from a database process might not. The use of ETag allows a MUD controller to more efficiently poll for changes in the file.

A manufacturer should also serve MUD files with an HTTP Max-Age header as per Section 5.2.2.8 of [RFC7234].

The MUD controller should take the Max-Age as an indication of when to next poll for updates to the MUD file. Values of less than 1 hour, or more than 1 month should be considered out of range, and clamped into the range (1 hour, 1 month).

MUD controllers SHOULD add some random jitter to the timing of their requests. MUD controllers MAY use a single HTTP(S)/1.1 connection to retrieve all resources at the same destination.

7. Privacy Considerations

The MUD URL could contain sensitive information such as the model number and even firmware revision numbers. Thus, the MUD URL may identify the make, model and revision of a device.

[RFC8520] already identifies this privacy concern, and suggests use of TLS so that the HTTP requests that retrieve the MUD file do not divulge that level of detail.

The requirement for the MUD controller to poll for changes to MUD files results in multiple interactions between the MUD controller and the manufacturer whereas a more naive implementation might only interact once. Even if HTTPS used, an observer of the traffic to that manufacturer will be revealing, and [RFC8520] goes on to suggest use of a proxy as well.

8. IANA Considerations

This document makes no requests to IANA.

9. Security Considerations

Prior to the standardization of the process in this document, if a device was infiltrated by malware, and said malware wished to make accesses beyond what the current MUD file allowed, the malware would have to:

1. arrange for an equivalent MUD file to be visible somewhere on the Internet

2. depend upon the MUD controller either not checking signatures, or

3. somehow get the manufacturer to sign the alternate MUD file

4. announce this new URL via DHCP or LLDP, updating the MUD controller with the new permissions.

One way to accomplish (3) is to leverage the existence of MUD files created by the manufacturer for different classes of devices. Such files would already be signed by the same manufacturer, eliminating the need to spoof a signature.

With the standardization of the process in this document, then the attacker can no longer point to arbitrary MUD files in step 4, but can only make use of MUD files that the manufacturer has already provided for this device.

Manufacturers are advised to maintain an orderly layout of MUD files in their web servers, with each unique product having its own directory/pathname.

The process described updates only MUD controllers and the processes that manufacturers use to manage the location of their MUD files.

A manufacturer which has not managed their MUD files in the way described here can deploy new directories of per-product MUD files, and then can update the existing MUD files in place to point to the new URLs using the MUD-URL attribute.

There is therefore no significant flag day: MUD controllers may implement the new policy without significant concern about backwards compatibility.

9.1. Updating files vs Updating MUD URLs

Device developers need to consider whether to make a change by updating a MUD file, or updating the MUD URL.

MUD URLs can only be updated by shipping a new firmware. It is reasonable to update the MUD URL whenever a new firmware release causes new connectivity to be required. The updated mechanism defined in this document makes this a secure operation, and there is no practical limitation on the number of files that a web server can hold.

In place updates to a MUD file should be restricted to cases where it turns out that the description was inaccurate: a missing connection, an inadvertent one authorized, or just incorrect information.

Developers should be aware that many enterprise websites use outsourced content distribution networks, and MUD controllers are likely to cache files for some time. Changes to MUD files will take some time to propagate through the various caches. An updated MUD URL will however, not experience any cache issues, but can not be deployed with a firmware update.

10. References

10.1. Normative References

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

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

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

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

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

10.2. Informative References

[boycrieswolf]
"The Boy Who Cried Wolf", 18 January 2020, <https://fablesofaesop.com/the-boy-who-cried-wolf.html>.

[boywolfinfosec]
"Security Alerts - A Case of the Boy Who Cried Wolf?", 18 January 2020, <https://www.infosecurity-magazine.com/opinions/security-alerts-boy-cried-wolf/>.

[falsemalware]
"False malware alerts cost organizations $1.27M annually, report says", 18 January 2020, <https://www.scmagazine.com/home/security-news/false-malware-alerts-cost-organizations-1-27m-annually-report-says/>.

[I-D.ietf-opsawg-mud-tls]
Reddy.K, T., Wing, D., and B. Anderson, "Manufacturer Usage Description (MUD) (D)TLS Profiles for IoT Devices", Work in Progress, Internet-Draft, draft-ietf-opsawg-mud-tls-18, 23 August 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-mud-tls-18>.

[I-D.jimenez-t2trg-mud-coap]
Jimenez, J., "Using MUD on CoAP environments", Work in Progress, Internet-Draft, draft-jimenez-t2trg-mud-coap-00, 9 March 2020, <https://datatracker.ietf.org/doc/html/draft-jimenez-t2trg-mud-coap-00>.

[IEEE_802.1AR-2018]
IEEE, "IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity", IEEE 802.1AR-2018, DOI 10.1109/IEEESTD.2018.8423794, August 2018, <https://doi.org/10.1109/IEEESTD.2018.8423794>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/info/rfc7232>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/info/rfc9111>.

[RFC9238] Richardson, M., Latour, J., and H. Habibi Gharakheili, "Loading Manufacturer Usage Description (MUD) URLs from QR Codes", RFC 9238, DOI 10.17487/RFC9238, May 2022, <https://www.rfc-editor.org/info/rfc9238>.

Appendix A. Appendices

Contributors

Jie Yang
Email: jay.yang@huawei.com

Tianqing Tang
Email: tangtianqing@huawei.com

Authors' Addresses

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca

Wei Pan
Huawei Technologies
Email: william.panwei@huawei.com

Eliot Lear
Cisco Systems
Email: lear@cisco.com