Workgroup: CoRE
Internet-Draft:
draft-ietf-core-transport-indication-05
Published: 19 March 2024
Intended Status: Standards Track
Expires: 20 September 2024
Authors: C. Amsüss M. S. Lenders
TU Dresden

CoAP Transport Indication

Abstract

The Constrained Application Protocol (CoAP, [RFC7252]) is available over different transports (UDP, DTLS, TCP, TLS, WebSockets), but lacks a way to unify these addresses. This document provides terminology and provisions based on Web Linking [RFC8288] to express alternative transports available to a device, and to optimize exchanges using these.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/.

Source for this draft and an issue tracker can be found at https://github.com/core-wg/transport-indication.

Status of This Memo

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) provides transports mechanisms (UDP and DTLS since [RFC7252], TCP, TLS and WebSockets since [RFC8323]), with some additional being used in LwM2M [lwm2m] and even more being explored ([I-D.bormann-t2trg-slipmux], [I-D.amsuess-core-coap-over-gatt]). These are mutually incompatible on the wire, but CoAP implementations commonly support several of them, and proxies can translate between them.

CoAP currently lacks a way to indicate which transports are available for a given resource, and to indicate that a device is prepared to serve as a proxy; this document solves both by introducing the "has-proxy" terminology to Web Linking [RFC8288] that expresses the former through the latter. The additional "has-unique-proxy" term is introduced to negate any per-request overhead that would otherwise be introduced in the course of this.

CoAP also lacks a unified scheme to label a resource in a transport-independent way. This document does *not* attempt to introduce any new scheme here, or raise a scheme to be the canonical one. Instead, each host or application can pick a canonical address for its resources, and advertise other transports in addition.

1.1. Terminology

Readers are expected to be familiar with the terms and concepts described in CoAP [RFC7252] and link format ([RFC6690] (or, equivalently, web links as described in [RFC8288]).

Same-host proxy: A CoAP server that accepts forward proxy requests (i.e., requests carrying the Proxy-Scheme option) exclusively for

URIs that it is also the authoritative server for is defined as a "same-host proxy".

The distinction between a same-host and any other proxy is only relevant on a practical, server-implementation and illustrative level; this specification does not use the distinction in normative requirements, and clients need not make the distinction at all.

hosts: The verb "to host" is used here in the sense of the link relation of the same name defined in [RFC6690].

For resources discovered via CoAP's discovery interface, a hosting statement is typically provided by the defaults implied by [RFC6690] where a link like </sensor/temp> is implied to have the relation "hosts" and the anchor /, such that a statement "coap://hostname hosts coap://hostname/sensor/temp" is implied in the link.

The link relation has been occasionally used with different interpretations, which ascribe more meaning to the term than it has in its definition. In particular,

* the "hosts" relation can not be inferred merely by two URIs having the same scheme, host and port (and vice versa), and

* the "hosts" relation on its own does not make any statement about the physical devices that hold the resource's representation.

[TBD: The former could probably still be used without too many ill effects; but things might also get weird when a dynamic resource created with one transport from use with another transport unless explicitly cleared.

Whether or not "to host" is used exclusively along the "hosts" relation or using the more generic same-start-of-URI sense is the largest open issue in this document.]

For the purpose of this document, "hosting" is used in a transitive way: If A hosts B and B hosts C, it is implied that A hosts C.

[TBD: It may make sense for many other relations to imply "hosts", e.g. any relations that occur in a pub-sub context, but that'd need further consideration.]

When talking of proxy requests, this document only talks of the Proxy-Scheme option. Given that all URIs this is usable with can be expressed in decomposed CoAP URIs, the need for using the Proxy-URI

option should never arise. The Proxy-URI option is still equivalent to the decomposed options, and can be used if the server supports it.

1.1.1. Using URIs to identify transport endpoints

The URI coap://[2001:db8::1] identifies a particular resource, possibly a "welcome" text. It is, colloquially, also used to identify the combination of a CoAP transport and the transport specific details.

For precision, this document uses the term "the transport address indicated by (a URI)" to refer to the transport and its details (in the example, CoAP over UDP with an IPv6 address and the default port), but otherwise no big deal is made of it.

The transport indicated by a URI is not only influenced by the URI scheme, but also by the authority component. The transports and resolution mechanisms currently specified make little use of this possibility, mainly because the most prominent resolution mechanism (SVCB records) has not been avaialble when [RFC8323] was published (see also Appendix E), end because it can not be expressed in IP literals (see Appendix F).

When the resolution mechanism used for a registered name authority component yields multiple addresses, all of those are possible ways to interact with the resource. The resolution mechanism or other underlying transport can give guidance on how to find the best usable one. With the currently specified transports and resolution mechanisms, the most prominent example of making use of that information is applying [RFC8305]'s Happy Eyeballs mechanism to establish a TCP connection when a name resolves to both IPv4 and IPv6 addresses,

1.1.1.1. Paths in URIs that indicate transport addresses

For the CoAP schemes (coap, coaps, coap+tcp, coaps+tcp, coap+ws, coaps+ws), URIs indicating a transport are always given with an empty path (which under their URI normalization rules is equivalent to a path containing a single slash). For the coap and coap+tcp schemes, URIs with different host names can indicate the same transport as long as the names resolve to the same addresses. For the others, the given host name informs the name set in TLS's Server Name Indication (SNI) and/or the host sent in the "Host" header of the underlying HTTP request.

If an update to this document extends the list, for new schemes it might be allowed to have paths, queries or fragment identifiers present in the URI indicating the transport address. No guidance can be given here for these, as no realistic example is known. (Note

that while the coap+ws scheme does use the well-known path /.well-known/coap internally, that is used purely on the HTTP side, and not part of the CoAP URI, not even for indicating the transport address). It is conceivable that a path such as the /.well-known/coap of CoAP-over-WebSockets would be indicated in an SVCB discovery's parameters similar to dohpath. This does not immediately help with the question of whether a URI indicating a transport can have a path, though. --CA

1.1.1.2. Existing use

A similar concept is used in [I-D.ietf-core-observe-multicast-notifications] (expressed as pieces of its tp_info parameter), but not expressed with URIs yet. As that document migrates towards using CRIs ([I-D.ietf-core-href]), it is expected that its transport addresses coincide with the URIs (CRIs, equivalently) indicating a transport.

URIs indicating a transport are especially useful when talking about proxies; this use is aligned with the way they are expressed in the conventional environment variables http_proxy etc. [noproxy]. Furthermore, URIs processing is widespread in CoAP systems, and when that changes (e.g. through the introduction of [I-D.ietf-core-href]), URIs indicating a transport will still be efficient to encode. And last but not least, it lines up well with the colloquial identity mentioned above. (An alternative would be using a dedicated naming scheme, say, transport:coap:device.example.com:port, but that would needlessly introduce implementation complexity).

Note that this mechanism can only used with proxies that use CoAP's native address indication mechanisms. Proxies that perform URI mapping (as described in Section 5 of [RFC8075], especially using URI templates) are not supported in this document.

[TBD: Do we want to extend this to HTTP proxies? Probably just not, and if so, only to those that can just take coap://... for a URI.]

1.2. Goals

This document introduces provisions for the seamless use of different transport mechanisms for CoAP. Combined, these provide:

1. Enablement: Inform clients of the availability of other transports of servers.

2. No Aliasing: Any URI aliasing must be opt-in by the server. Any defined mechanisms must allow applications to keep working on the canonical URIs given by the server.

3. Optimization: Do not incur per-request overhead from switching transports. This may depend on the server's willingness to create aliased URIs.

4. Proxy usability: All information provided must be usable by aware proxies to reduce the need for duplicate cache entries.

5. Proxy announcement: Allow third parties to announce that they provide alternative transports to a host.

For all these functions, security policies must be described that allow the client to use them as securely as the original transport.

This document will not concern itself with changes in transport availability over time, neither in causing them ("Please take up your TCP interface, I'm going to send a firmware update") nor in advertising their availability in advance. Hosts whose transport's availability changes over time can utilize any suitable mechanism to keep client updated, such as placing a suitable Max-Age value on their resources or having them observable.

2. Indicating alternative transports

While CoAP can set the authority component of the requested URI in all requests (by means of Uri-Host and Uri-Port), setting the scheme of a requested URI (by means of Proxy-Scheme) makes the request implicitly a proxy request. However, this needs to be of only little practical concern: Any device can serve as a proxy for itself (a "same-host proxy") by accepting requests that carry the Proxy-Scheme option. Section 5.7.2 of [RFC7252] already mandates that a proxy recognize its own addresses. A minimal same-host proxy supports only those and respond with 5.05 (Proxying Not Supported). In many cases (precisely: on hosts that alias their resources across transports), this is equivalent to ignoring the Proxy-Scheme option in that request.

A server can advertise a recommended proxy by serving a Web Link with the "has-proxy" relation to a URI indicating its transport address. In particular (and that is a typical case), it can indicate its own transport address on an alternative transport when implementing same-host proxy functionality.

The semantics of a link from S to P with relations has-proxy ("S has-proxy P", <P>;rel=has-proxy;anchor="S") are that for any resource R hosted on S ("S hosts R"), the proxy with the transport address indicated by P can be used to obtain R.

2.1. Example

A constrained device at the address 2001:db8::1 that supports CoAP over TCP in addition to CoAP can self-describe like this:

```
Req: to [ff02::fd]:5683 on UDP
Code: GET
Uri-Path: /.well-known/core
Uri-Query: if=tag:example.com,sensor

Res: from [2001:db8::1]:5683
Content-Format: application/link-format
Payload:
</sensors/temp>;if="tag:example.com,sensor",
<coap+tcp://[2001:db8::1]>;rel=has-proxy;anchor="/"


Req: to [2001:db8::1]:5683 on TCP
Code: GET
Proxy-Scheme: coap
Uri-Path: /sensors/temp
Observe: 0

Res: 2.05 Content
Observe: 0
Payload:
39.1°C
```

Figure 1: Discovery and follow-up request through a has-proxy relation

Note that generating this discovery file needs to be dynamic based on its available addresses; only if queried using a link-local source address, the server may also respond with a link-local address in the authority component of the proxy URI.

Unless the device makes resources discoverable at coap+tcp://[2001:db8::1]/.well-known/core or another discovery mechanism, clients may not assume that coap+tcp://[2001:db8::1]/sensors/temp is a valid resource (let alone is equivalent to the other resource on the same path). The server advertising itself like this may reject any request on CoAP-over-TCP unless it contains a Proxy-Scheme option.

Clients that want to access the device using CoAP-over-TCP would send a request by connecting to 2001:db8::1 TCP port 5683 and sending a GET with the options Proxy-Scheme: coap, no Uri-Host or -Port options (utilizing their default values), and the Uri-Paths "sensors" and "temp".

2.2. Security context propagation

Any security requirements posed by a server or client application on a CoAP request MUST be applied independently of the transport that is used to perform the request. If a transport can not be used to satisfy the requirements, it is ineligible for use with the request (from a client's point of view), and unauthorized (from a server's point of view).

If the requirements contain transport layer security, the proxy needs to present the credentials required of the server to the client, and those of the client to the server; this is only practical when the proxy is a same-host proxy.

Some applications have requirements exceeding the requirements of a secure connection, e.g., (explicitly or implicitly) requiring that name resolution happen through a secure process and packets are only routed into networks where it trusts that they will not be intercepted on the path to the server. Such applications need to extend their requirements to the source of the has-proxy statement; a sufficient (but maybe needlessly strict) requirement is to only follow has-proxy statements that are part of the same resource that advertises the link currently being followed. Section Section 7.2 adds further considerations.

2.3. Choice of transports

It is up to the client whether to use an advertised proxy transport, or (if multiple are provided) which to pick.

Links to proxies may be annotated with additional metadata that may help guide such a choice; defining such metadata is out of scope for this document.

Clients MAY switch between advertised transports as long as the document describing them is fresh; they may even do so per request. (For example, they may perform individual requests using CoAP-over-UDP, but choose CoAP-over-TCP for requests with large expected responses). When the describing document approaches expiry, the client can use the representation's ETag to efficiently renew its justification for using the alternative transport.

2.4. Selection of a canonical origin

While a server is at liberty to provide the same resource independently on different transports (i.e. to create aliases), it may make sense for it to pick a single scheme and authority under which it announces its resources. Using only one address helps proxies keep their caches efficient, and makes it easier for clients to avoid exploring the same server twice from different angles.

When there is a predominant scheme and authority through which an existing service is discovered, it makes sense to use these for the canonical addresses.

Otherwise, it is suggested to use the coap or coaps scheme (given that these are the most basic and widespread ones), and the most stable usable name the host has.

2.4.1. Unreachable canonical origin addresses

For devices that are not generally reachable at a stable address, it may make sense to use a scheme and authority as the canonical address that can not actually be dereferenced.

The registered names available for that purpose depend on the locally defined host or service name registry. When the Domain Name System (DNS) is used, such names would not be associated with any A or AAAA records (but may still use, for example, TLSA records).

Such URIs are *only* usable to clients that discover a suitable proxy along with the URI, and which can place sufficient trust in that proxy.

2.5. Advertisement through a Resource Directory

In the Resource Directory specification [rfc9176], protocol negotiation was anticipated to use multiple base values. This approach was abandoned since then, as it would incur heavy URI aliasing.

Instead, devices can submit their has-proxy links to the Resource Directory like all their other metadata.

A client performing resource lookup can ask the RD to provide available (same-host-)proxies in a follow-up request by asking for ?anchor=<the-discovered-host>&rel=has-proxy. The RD may also volunteer that information during resource lookups even though the has-proxy link itself does not match the search criteria.

[

It may be useful to define RD parameters for use with lookup here, which'd guide which available proxies to include. For example, asking ?if=tag:example.com,sensor&proxy-links=tcp could give as a result:

<coap://[2001:db8::1]/s>;rt=tag:example.com,sensor,<coap+tcp://[2001:db8::1]/>;rel=has-proxy;anchor="coap://[2001:db8::1]/"

This is similar to the extension suggested in Section 5 of [I-D.amsuess-core-resource-directory-extensions].

]

3. Elision of Proxy-Scheme and Uri-Host

A CoAP server may publish and accept multiple URIs for the same resource, for example when it accepts requests on different IP addresses that do not carry a Uri-Host option, or when it accepts requests both with and without the Uri-Host option carrying a registered name. Likewise, the server may serve the same resources on different transports. This makes for efficient requests (with no Proxy-Scheme or Uri-Host option), but in general is discouraged [aliases].

To make efficient requests possible without creating URI aliases that propagate, the "has-unique-proxy" specialization of the has-proxy relation is defined.

If a proxy is unique, it means that requests arriving at the proxy are treated the same no matter whether the scheme, authority and port of the link context are set in the Proxy-Scheme, Uri-Host and Uri-Port options, respectively, or whether all of them are absent.

[The following two paragraphs are both true but follow different approaches to explaining the observable and implementable behavior; it may later be decided to focus on one or the other in this document.]

While this creates URI aliasing in the requests as they are sent over the network, applications that discover a proxy this way should not "think" in terms of these URIs, but retain the originally discovered URIs (which, because Cool URIs Don't Change[cooluris], should be long-term usable). They use the proxy for as long as they have fresh knowledge of the has-(unique-)proxy statement.

In a way, advertising has-unique-proxy can be viewed as a description of the link target in terms of SCHC [I-D.ietf-lpwan-coap-static-context-hc]: In requests to that target, the link source's scheme and host are implicitly present.

While applications retain knowledge of the originally requested URI (even if it is not expressed in full on the wire), the original URI is not accessible to caches both within the host and on the network (for the latter, see Section 5). Thus, cached responses to the canonical and any aliased URI are mutually interchangeable as long as both the response and the proxy statement are fresh.

A client MAY use a unique-proxy like a proxy and still send the Proxy-Scheme and Uri-Host option; such a client needs to recognize both relation types, as relations of the has-unique-proxy type are a specialization of has-proxy and typically don't carry the latter (redundant) annotation. [To be evaluated -- on one hand, supporting it this way means that the server needs to identify all of its addresses and reject others. Then again, is a server that (like many now do) fully ignore any set Uri-Host correct at all?]

Example:

```
Req: to [ff02::fd]:5683 on UDP
Code: GET
Uri-Path: /.well-known/core
Uri-Query: if=tag:example.com,sensor

Res: from [2001:db8::1]:5683
Content-Format: application/link-format
Payload:
</sensors/>;if="tag:example.com,collection",
<coaps+ws://[2001:db8::1]>;rel=has-unique-proxy;anchor="/"


Req: to [2001:db8::1] via WebSockets over HTTPS
Code: GET
Uri-Path: /sensors/

Res: 2.05 Content
Content-Format: application/link-format
Payload:
</sensors/temperature>;if="tag:example.com,sensor"
```

Figure 2: Follow-up request through a has-unique-proxy relation. Compared to the last example, 5 bytes of scheme indication are saved during the follow-up request.

It is noteworthy that when the URI reference /sensors/temperature is resolved, the base URI is coap://device0815.example.com and not its coaps+ws counterpart -- as the request is still for that URI, which both the client and the server are aware of. However, this detail is of little practical importance: A simplistic client that uses coaps+ws://device0815.proxy.rd.example.com as a base URI will still arrive at an identical follow-up request with no ill effect, as long as it only uses the wrongly assembled URI for dereferencing resources, the security context is the same, the state is kept no longer than the has-unique-proxy statement is fresh, and it does not (for example) pass the URI on to other devices.

3.1. Impact on caches

[This section is written with the "there is implied URI aliasing" mindset; it should be possible to write it with the "compression" mindset as well (but there is no point in having both around in the document at this time).

It is also slightly duplicating, but also more detailed than, the brief note on the topic in Section 5]

When a node that performs caching learns of a has-unique-proxy statement, it can utilize the information about the implied URI aliasing: Requests to resources hosted by S can be answered with cached entries from P (because by the rules of has-unique-proxy a request can be crafted that is sent to P for which a fresh response is available). The inverse direction (serving resources whose URI "starts with" P from a cached request that was sent to S) is harder to serve because it additionaly requires a fresh statement that "S hosts R" for the matching resource R.

3.2. Using unique proxies securely

The elision of the host name afforded by the unique-proxy relation is only possible if the required security mechanisms verify the scheme and host of the server.

This is given for OSCORE based mechanisms, where "unprotected message fields (including Uri-Host [...]) MUST not lead to an OSCORE message becoming verified".

With TLS based security mechanisms, name and scheme can not be completely elided in general. While the use of the SNI HostName field sets the default Uri-Host already, the scheme still needs to be sent in a Proxy-Scheme option to satisfy the requirement of Section 2.2.

[It may be possible to relax this requirement if the host publishes a *trustworthy* statement about serving the same content on all schemes; however, no urgent need for this optimization is currently known that warrants the extra scrutiny.]

4. Third party proxy services

A server that is aware of a suitable cross proxy may use the has-proxy relation to advertise that proxy. If the transport used towards the proxy provides name indication (as CoAP over TLS or WebSockets does), or by using a large number of addresses or ports, it can even advertise a (more efficient) has-unique-proxy relation. This is particularly interesting when the advertisements are made available across transports, for example in a Resource Directory.

How the server can discover and trust such a proxy is out of scope for this document, but generally involves the same kind of links. In particular, a server may obtain a link to a third party proxy from an administrator as part of its configuration.

The proxy may advertise itself without the origin server's involvement; in that case, the client needs to take additional care (see Section 7.2).

```
Req: GET http://rd.example.com/rd-lookup?if=tag:example.com,sensor

Res:
Content-Format: application/link-format
Payload:
<coap://device0815.example.com/sensors/>;if="tag:example.com,collection"
<coap+wss://device0815.proxy.rd.example.com>;rel=has-unique-proxy;anchor


Req: to device0815.proxy.rd.example.com on WebSocket
Host (indicated during upgrade): device0815.proxy.rd.example.com
Code: GET
Uri-Path: /sensors/

Res: 2.05 Content
Content-Format: application/link-format
Payload:
</sensors/temperature>;if="tag:example.com,sensor"
```

Figure 3: HTTP based discovery and CoAP-over-WS request to a CoAP resource through a has-unique-proxy relation

4.1. Generic proxy advertisements

A third party proxy may advertise its availability to act as a proxy for arbitrary CoAP requests. This use is not directly related to the transport indication in other parts of this document, but sufficiently similar to warrant being described in the same document.

The resource type "TBDcore.proxy" can be used to describe such a proxy.

```
Req: GET coap://[fe80::1]/.well-known/core?rt=TBDcore.proxy

Res:
Content-Format: application/link-format
Payload:
<>;rt=TBDcore.proxy

Req: to [fe80::1] via CoAP
Code: GET
Proxy-Scheme: http
Uri-Host: example.com
Uri-Path: /motd
Accept: text/plain

Res: 2.05 Content
Content-Format: text/plain
Payload:
On Monday, October 25th 2021, there is no special message of the day.
```

Figure 4: A CoAP client discovers that its border router can also serve as a proxy, and uses that to access a resource on an HTTP server.

The considerations of Section 7.2 apply here.

A generic advertised proxy is always a forward proxy, and can not be advertised as a "unique" proxy as it would lack information about where to forward.

A proxy may be limited in the URIs it can service, for technical reasons (e.g. when none of the URI's transports are supported by the server) or for policy reasons (only accessing servers inside an organizational structure). Future documents (or versions of this document) may add target attributes that allow specifying the capabilities of a proxy. [An earlier version of this document contained a proxy-schemes attribute. This was discontinued because it could already not express whether a proxy could access IPv4 or IPv6 peers, and because the use of schemes is becoming less useful given the new recommendation of incorporating details from registered name resolution into the transport selection.]

The use of a generic proxy can be limited to a set of devices that have permission to use it. Clients can be allowed by their network address if they can be verified, or by using explicit client authentication using the methods of [I-D.tiloca-core-oscore-capable-proxies].

5. Client picked proxies

This section is purely informative, and serves to illustrate that the mechanisms introduced in this document do not hinder the continued use of existing proxies.

When a resource is accessed through an "actual" proxy (i.e., a host between the client and the server, which itself may have a same-host proxy in addition to that), the proxy's choice of the upstream server is originally (i.e., without the mechanisms of this document) either configured (as in a "chain" of proxies) or determined by the request URI (where a proxy picks CoAP over TCP and resolves the given name for a request aimed at a coap+tcp URI).

A proxy that has learned, by active solicitation of the information or by consulting links in its cache, that the requested URI is available through a (possibly same-host) proxy, may use that information in choosing the upstream transport, to correct the URI associated with a cached response, and to use responses obtained through one transport to satisfy requests on another.

For example, if a host at coap://h1.example.com has advertised </res>,<coap+tcp://h1.example.com>;rel=has-proxy;anchor="/", then a proxy that has an active CoAP-over-TCP connection to h1.example.com can forward an incoming request for coap://h1.example.com/res through that CoAP-over-TCP connection with a suitable Proxy-Scheme and Uri-Host on that connection.

If the host had marked the proxy point as <coap+tcp://h1.example.com>;rel=has-unique-proxy instead, then the proxy could elide the Proxy-Scheme and Uri-Host options, and would (from the original CoAP caching rules) also be allowed to use any fresh cache representation of coap+tcp://h1.example.com/res to satisfy requests for coap://h1.example.com/res.

A client that uses a forward proxy and learns of a different proxy advertised to access a particular resource will not change its behavior if its original proxy is part of its configuration. If the forward proxy was only used out of necessity (e.g., to access a resource whose indicated transport not supported by the client) it can be practical for the client to use the advertised proxy instead.

6. Guidance to upcoming transports

When new transports are defined for CoAP, it is recommended to use the "coap" scheme (or "coaps" for TLS based transports).

If the transport's identifiers are IP based and have identifiers typically resolved through DNS, authors of new transports are encouraged to specify Service Binding records ([RFC9460]) for CoAP

(possibly taking inspiration from Appendix E), and if IP literals are relevant to the transport, to follow up on Appendix F.

If the transport's native identifiers are compatible with the structure of the authority component of a URI, those identifiers can be used as an authority as-is. To help the host decide the resolution mechanism, it may be helpful to register a subdomain of .arpa as described in [rfc3172]. The guidence for users is to never attempt to resolve such a name, and for the zone's implementation is to return NXDOMAIN unconditionally.

If the transport's native identifiers are incompatible with that structure (e.g. because they contain colons), the document may define some transformation.

If a transport's native identifiers are only local, the zone .alt [rfc9476] may be used instead.

For example, CoAP over GATT [I-D.amsuess-core-coap-over-gatt] removes the colons from Bluetooth Low Energy MAC addresses like 00:11:22:33:44:55 and combines them into authority compoennts such as 001122334455.ble.arpa. Slipmux [I-D.bormann-t2trg-slipmux] might use the locally significant device name /dev/ttyUSB0 as coap://ttyUSB0.dev.alt/.

URIs created from such names may not indicate the protocol uniquely: Additional transports specified later may also provide CoAP services for the same name. In the sense of Section 1.1.1, both transport would be identified by that URI. That is not an issue as long as the protocols underneath the CoAP transport provide a means of advertising the precise protocol used. For example, a hypothetical CoAP transport for BLE that is not GATT based can be selected for the same scheme and authority based on data in the BLE advertisement.

7. Security considerations

7.1. Security context propagation

Clients need to strictly enforce the rules of Section 2.2. Failure to do so, in particular using a thusly announced proxy based on a certificate that attests the proxy's name, would allow attackers to circumvent the client's security expectation.

When security is terminated at proxies (as is in DTLS and TLS), a third party proxy can usually not satisfy this requirement; these transports are limited to same-host proxies.

7.2. Traffic misdirection

Accepting arbitrary proxies, even with security context propagation performed properly, would allow attackers to redirect traffic through systems under their control. Not only does that impact availability, it also allows an attacker to observe traffic patterns.

This affects both OSCORE and (D)TLS, as neither protect the participants' network addresses.

Other than the security context propagation rules, there are no hard and general rules about when an advertised proxy is a suitable candidate. Aspects for consideration are:

* When no direct connection is possible (e.g. because the resource to be accessed is served as coap+tcp and TCP is not implemented in the client, or because the resource's host is available on IPv6 while the client has no default IPv6 route), using a proxy is necessary if complete service disruption is to be avoided.

  While an adversary can cause such a situation (e.g. by manipulating routing or DNS entries), such an adversary is usually already in a position to observe traffic patterns.

* A proxy advertised by the device hosting the resource to be accessed is less risky to use than one advertised by a third party.

  The /.well-known/core resource is regarded as a source of authoritative information on the endpoint's CoAP related metadata, and can be queried early in the discovery process, or queried for verification (with filtering applied) after discovery through an RD. Other resources may be less trustworthy as they may be controlled by entities not trusted with the endpoint's traffic.

Appendix D describes an extension to [I-D.ietf-lake-edhoc] by which the client can verify that the proxy used by the client is recognized by the server. This is similar to querying /.well-known/core for any proxies advertised there, but happens earlier in the connection establishment, and leaves the decision whether the proxy is legitimate to the server.

It only conveys information about the URI of the proxy. The mapping of any host name inside it to an IP address, or of an IP address to a routing decision, is left to the security mechanisms of the respective layers.

7.3. Protecting the proxy

A widely published statement about a host's availability as a proxy can cause many clients to attempt to use it.

This is mitigated in well-behaved clients by observing the rate limits of [RFC7252], and by ceasing attempts to reach a proxy for the Max-Age of received errors.

Operators can further limit ill-effects by ensuring that their client systems do not needlessly use proxies advertised in an unsecured way, and by providing own proxies when their clients need them.

8. IANA considerations

8.1. Link Relation Types

IANA is asked to add two entries into the Link Relation Type Registry last updated in [RFC8288]:

Relation Name	Description	Reference
has-proxy	The link target can be used as a proxy to reach the link context.	RFCthis
has-unique-proxy	Like has-proxy, and using this proxy implies scheme and host of the target.	RFCthis

Table 1: New Link Relation types

8.2. Resource Types

IANA is asked to add an entry into the "Resource Type (rt=) Link Target Attribute Values" registry under the Constrained RESTful Environments (CoRE) Parameters:

[The RFC Editor is asked to replace any occurrence of TBDcore.proxy with the actually registered attribute value.]

Attribute Value: core.proxy

Description: Forward proxying services

Reference: [this document]

9. References

9.1. Normative References

[RFC7252]

Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/rfc/rfc8288>.

9.2. Informative References

[aliases] W3C, "Architecture of the World Wide Web, Section 2.3.1 URI aliases", n.d., <https://www.w3.org/TR/webarch/#uri-aliases>.

[cooluris] BL, T., "Cool URIs don't change", n.d., <https://www.w3.org/Provider/Style/URI>.

[evossl] Baier, E., "The Evolution of SSL and TLS", 2 February 2015, <https://www.digicert.com/blog/evolution-of-ssl>.

[I-D.amsuess-core-coap-over-gatt]
Amsüss, C., "CoAP over GATT (Bluetooth Low Energy Generic Attributes)", Work in Progress, Internet-Draft, draft-amsuess-core-coap-over-gatt-05, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-amsuess-core-coap-over-gatt-05>.

[I-D.amsuess-core-resource-directory-extensions]
Amsüss, C., "CoRE Resource Directory Extensions", Work in Progress, Internet-Draft, draft-amsuess-core-resource-directory-extensions-10, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-amsuess-core-resource-directory-extensions-10>.

[I-D.amsuess-t2trg-rdlink]
Amsüss, C., "rdlink: Robust distributed links to constrained devices", Work in Progress, Internet-Draft, draft-amsuess-t2trg-rdlink-01, 23 September 2019, <https://datatracker.ietf.org/doc/html/draft-amsuess-t2trg-rdlink-01>.

[I-D.bormann-t2trg-slipmux] Bormann, C. and T. Kaupat, "Slipmux: Using an UART interface for diagnostics, configuration, and packet transfer", Work in Progress, Internet-Draft, draft-bormann-t2trg-slipmux-03, 4 November 2019,

<https://datatracker.ietf.org/doc/html/draft-bormann-t2trg-slipmux-03>.

[I-D.ietf-core-href] Bormann, C. and H. Birkholz, "Constrained Resource Identifiers", Work in Progress, Internet-Draft, draft-ietf-core-href-14, 9 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-core-href-14>.

[I-D.ietf-core-observe-multicast-notifications] Tiloca, M., Höglund, R., Amsüss, C., and F. Palombini, "Observe Notifications as CoAP Multicast Responses", Work in Progress, Internet-Draft, draft-ietf-core-observe-multicast-notifications-08, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-core-observe-multicast-notifications-08>.

[I-D.ietf-lake-edhoc] Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-23, 22 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-23>.

[I-D.ietf-lpwan-coap-static-context-hc] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-lpwan-coap-static-context-hc-19, 8 March 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-lpwan-coap-static-context-hc-19>.

[I-D.lenders-core-dnr] Lenders, M. S., Amsüss, C., Schmidt, T. C., and M. Wählisch, "Discovery of Network-designated CoRE Resolvers", Work in Progress, Internet-Draft, draft-lenders-core-dnr-00, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-lenders-core-dnr-00>.

[I-D.silverajan-core-coap-protocol-negotiation] Silverajan, B. and M. Ocak, "CoAP Protocol Negotiation", Work in Progress, Internet-Draft, draft-silverajan-core-coap-protocol-negotiation-09, 2 July 2018, <https://datatracker.ietf.org/doc/html/draft-silverajan-core-coap-protocol-negotiation-09>.

[I-D.tiloca-core-oscore-capable-proxies] Tiloca, M. and R. Höglund, "OSCORE-capable Proxies", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-capable-proxies-07, 10

July 2023, <https://datatracker.ietf.org/doc/html/draft-tiloca-core-oscore-capable-proxies-07>.

[lwm2m] OMA SpecWorks, "White Paper – Lightweight M2M 1.1", n.d., <https://omaspecworks.org/white-paper-lightweight-m2m-1-1/>.

[noproxy] Hu, S., "We need to talk: Can we standardize NO_PROXY?", 27 January 2021, <https://about.gitlab.com/blog/2021/01/27/we-need-to-talk-no-proxy/>.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/rfc/rfc2616>.

[rfc3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <https://www.rfc-editor.org/rfc/rfc3172>.

[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/rfc/rfc3986>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/rfc/rfc5952>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/rfc/rfc6690>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/rfc/rfc6698>.

[RFC7838] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <https://www.rfc-editor.org/rfc/rfc7838>.

[RFC8075] Castellani, A., Loreto, S., Rahman, A., Fossati, T., and E. Dijk, "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)", RFC 8075, DOI 10.17487/RFC8075, February 2017, <https://www.rfc-editor.org/rfc/rfc8075>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/rfc/rfc8305>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/rfc/rfc8323>.

[rfc9176] Amsüss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/rfc/rfc9176>.

[RFC9460] Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, DOI 10.17487/RFC9460, November 2023, <https://www.rfc-editor.org/rfc/rfc9460>.

[rfc9476] Kumari, W. and P. Hoffman, "The .alt Special-Use Top-Level Domain", RFC 9476, DOI 10.17487/RFC9476, September 2023, <https://www.rfc-editor.org/rfc/rfc9476>.

[w3address] BL, T., "W3 address syntax: BNF", 29 June 1992, <http://info.cern.ch/hypertext/WWW/Addressing/BNF.html#43>.

Appendix A. Change log

Since draft-ietf-core-transport-indication-04:

* Not just the scheme, but also the authority value influences the transport selection.

- Add guidance section for new transports.

- Point out that registerd names already can fan out to different addresses.

* Rephrase and simplify security considerations, especially by limiting unique proxying for TLS.

* Add recommendation to new scheme authors to use "coap"/"coaps" and let the resolution process guide the selection.

- Remove proxy-schemes attribute from core.proxy because of its greatly reduced value.

* Update "Related work" appendix to cover SVCB instead of SRV records

* Rename to "Transport Indication", using "protocol" only for other protocols, in established phrases, or when referring to CoAP as a general protocol.

* Add note linking CoAP-over-WS's .well-known/coap to dohpath

* Remove OSCORE vs. unique-proxy open point

* EDHOC EAD: Describe response option content

* Editorial updates

Since draft-ietf-core-transport-indication-03:

* Added appendices on alternative history and Literals beyond IP addresses. The remaining document was not brought in sync with those new parts.

Since draft-ietf-core-transport-indication-02:

* Added EAD appendix, adjusted security considerations to match.

Since draft-ietf-core-transport-indication-01:

* Simplify same-host proxy behavior by referring to existing RFC7252 mandate.

* proxy-links= lookup: Refer to prior art.

Since draft-ietf-core-transport-indication-00:

* Add section on canonical URIs that are not necessarily reachable.

* Clarify that the the "hosts" relation is followed transitively.

* Cross reference with compatible multicast-notifications concept.

Since draft-amsuess-core-transport-indication-03:

* No changes (merely changing the name after WG adoption)

Since -02 (mainly processing reviews from Marco and Klaus):

* Acknowledge that 'coap://hostname/' is not the proxy but a URI that (in a particular phrasing) is used to stand in for the proxy's address (while it regularly identifies a resurce on the server)

* Security: Referencing traffic misdirection already in the first security block.

* Security: Add (incomplete) considerations for unique-proxy case.

* Narrow down "unique" proxy semantics to those properties used by the client, allowing unique proxies to be co-hosted with forward proxies.

* "Client picked proxies" clarified to merely illustrate how this is compatible with them.

* Use of "hosts" relation sharpened.

* Precision on how this does and does not consider changing transports.

* "Related work" section demoted to appendix.

* Add note on DTLS session resumption.

* Variable renaming.

* Various editorial fixes.

Since -01:

* Removed suggestion for generally trusted proxies; now stating that with (D)TLS, "a third party proxy can usually not satisfy [the security context propagation requirement]".

* State more clearly that valid cache entries for resources aliased through has-unique-proxy can be used.

* Added considerations for Multipath TCP.

* Added concrete suggestion and example for advertisement of general proxies.

*Added concrete suggestion for RD lookup extension that provides proxies.

*Minor editorial and example changes.

Since -00:

*Added introduction

*Added examples

*Added SCHC analogy

*Expanded security considerations

*Added guidance on choice of transport, and canonical addresses

*Added subsection on interaction with a Resource Directory

*Added comparisons with related work, including rdlink and DNS-SD sketches

*Added IANA considerations

*Added section on open questions

Appendix B. Related work and applicability to related fields

B.1. On HTTP

The mechanisms introduced here are similar to the Alt-Svc header of [RFC7838] in that they do not create different application-visible addresses, but provide dispatch through lower transport implementations.

In HTTP, different versions of the protocol (i.e., different transports) are distinguished using a protocol identifier equivalent to an ALPN. This works well because all relevant transports use transport layer security and thus can use ALPNs. In contrast, the currently specified CoAP transports predate ALPNs, and specified per-transport schemes, which enable the use of URIs that indicate transports.

To accommodate the message size constraints typical of CoRE environments, and accounting for the differences between HTTP headers and CoAP options, information is delivered once at discovery time.

Using the has-proxy and has-unique-proxy with HTTP URIs as the context is NOT RECOMMENDED; the HTTP provisions of the Alt-Svc header and ALPN are preferred.

B.2. Using DNS

DNS Service Binding resource records (SVCB RRs) described in [RFC9460] can carry many of the details otherwise negotiated using the proxy relations. In HTTP, they can be used in a way similar to Alt-Svc headers.

SVCB records were not specified when CoAP was specified for TCP, but could have been (see Appendix E).

If at any point SVCB records for CoAP are defined, name resolution produces a set of transport details that can be used immediately without the need for a has-proxy link. Explicit has-proxy links would still be relevant for third party advertised proxies.

B.3. Using names outside regular DNS

Names that are resolved through different mechanisms than DNS, or names which are defined within the scope of DNS but have no universally valid answers to A/AAAA requests, can be advertised using the relation types defined here and CoAP discovery.

In Figure 5, a server using a cryptographic name as described in [I-D.amsuess-t2trg-rdlink] is discovered and used.

```
Req: to [ff02::fd]:5683 on UDP
Code: GET
Uri-Path: /.well-known/core
Uri-Query: rel=has-proxy
Uri-Query: anchor=coap://nbswy3dpo5xxe3denbswy3dpo5xxe3de.ab.rdlink.arpa

Res: from [2001:db8::1]:5683
Content-Format: application/link-format
Payload:
<coap+tcp://[2001:db8::1]>;rel=has-unique-proxy;
  anchor="coap://nbswy3dpo5xxe3denbswy3dpo5xxe3de.ab.rdlink.arpa"


Req: to [2001:db8::1]:5683 on TCP
Code: GET
OSCORE: ...
Uri-Path: /sensors/temp
Observe: 0

Res: 2.05 Content
OSCORE: ...
Observe: 0
Payload:
39.1°C
```

Figure 5: Obtaining a sensor value from a local device with a global name

B.4. Multipath TCP

When CoAP-over-TCP is used over Multipath TCP and no Uri-Host option is sent, the implicit assumption is that there is aliasing between URIs containing any of the endpoints' addresses.

As these are negotiated within MPTCP, this works independently of this document's mechanisms. As long as all the server's addresses are equally reachable, there is no need to advertise multiple addresses that can later be discovered through MPTCP anyway. When advertisements are long-lived and there is no single more stable address, several available addresses can be advertised (independently of whether MPTCP is involved or not). If a client uses an address that is merely a proxy address (and not a unique proxy address), but during MPTCP finds out that the network location being accessed is actually an MPTCP alternative address of the used one, the client MAY forego sending of the Proxy-Scheme and Uri-Path option.

[This follows from multiple addresses being valid for that TCP connection; at some point we may want to say something about what that means for the default value of the Uri-Host option -- maybe

something like "has the default value of any of the associated addresses, but the server may only enable MPTCP if there is implicit aliasing between all of them" (similar to OSCORE's statement)?]

[TBD: Do we need a section analog to this that deals with (D)TLS session resumption in absence of SNI?]

Appendix C. Open Questions / further ideas

* Self-uniqueness:

  A host that wants to indicate that it doesn't care about Uri-Host can probably publish something like </>;rel=has-unique-proxy to do so.

  This'd help applications justify when they can elide the Uri-Host, even when no different transports are involved.

* Advertising under a stable name:

  If a host wants to advertise its host name rather than its IP address during multicast, how does it best do that?

  Options, when answering from 2001:db8::1 to a request to ff02::fd are:

  <coap://myhostname/foo>,...,
  <coap://[2001:db8::1]>;rel=has-unique-proxy;anchor="coap://myhostname"

  which is verbose but formally clear, and

  </foo>,...,
  <coap://[2001:db8::1]>;rel=has-unique-proxy;anchor="coap://myhostname"

  which is compatible with unaware clients, but its correctness with respect to canonical URIs needs to be argued by the client, in this sequence

  - understanding the has-unique-proxy line,

  - understanding that the request that went to 2001:db8::1 was really a Proxy-Scheme/Uri-Host-elided version of a request to coap://myhostname, and then

  - processing any relative reference with this new base in mind.

  (Not that it'd need to happen in software in that sequence, but that's the sequence needed to understand how the /foo here is really coap://myhostname/foo).

If CoRAL is used during discovery, a base directive or reverse relation to has-unique-proxy would make this easier.

Appendix D. EDHOC EAD for verifying legitimate proxies

This document sketches an extension to [I-D.ietf-lake-edhoc] that informs the server of the public address the client is using, allowing it to detect undesired reverse proxies.

[This section is immature, and written up as a discussion starting point. Further research into prior art is still necessary.]

The External Authorization Data (EAD) item with name "Proxy CRI", label TBD24, is defined for use with messages 1, 2 and 3.

A client can set this label in uncritical form, followed by a CRI ([I-D.ietf-core-href]) that is CBOR-encoded in a byte string as a CBOR sequence. The transport indicated by the URI is the proxy the client chose from information advertised about the server.

If a server can not determine its set of legitimate proxies, it ignores the option (as does any EDHOC implementation that is unaware of it).

If it recognizes the CRI as belonging to a legitimate proxy, it places the empty label in its non-critical form in the next message to confirm the proxy choice. Otherwise, it places the label in its critical form, either empty or containing a recommended CRI. The client may then decide to discontinue using the proxy, or to use more extensive padding options to sidestep the attack. Both the client and the server may alert their administrators of a possible traffic misdirection.

Appendix E. Alternative History: What if SVCB had been around before CoAP over TCP?

This appendix explores a hypothetical scenario in which Service Binding (SVCB, [RFC9460]) was around and supported before the controversial decision to establish the "coap+tcp" scheme. It serves to provide a fresh perspective of what parts are logically necessary, and to ease the exploration of how it may be used in the future.

E.1. Hypothetical retrospecification

CoAP is specified for several transports: CoAP over UDP, over DTLS, over TCP, over TLS and over (secure or insecure) WebSockets. URIs of all these are expressed using the "coap" or "coaps" scheme, depending on whether a (D)TLS connection is to be used. [It is currently unclear whether the two schemes should also be unified;

the rest of the text is left intentionally vague on that distinction.]

Any server providing CoAP services announces not only its address but also its SVCB Service Parameters, including at least one of alpn and coaptransfer.

For example, a host serving "coap://sensor.example.com" and "coaps://sensor.example.com" might have these records:

```
_coap.sensor.example.com IN SVCB 1 . alpn=coap,co
coaptransfer=tcp,udp port=61616 sensor.example.com IN AAAA
2001:db8::1
```

A client connecting to the server loops up the name's service parameters using its system's discovery mechanisms.

For example, if DNS is used, it obtains SVCB records for _coap.sensor.example.com, and receives the corresponding AAAA record either immediately from an SVCB aware resolver or through a second query. It learns that the service is available through CoAP-over-DTLS (ALPN "co"), CoAP-over-TLS (ALPN "coap"), or through unencrypted TCP or UDP, and that port 61616 needs to be used in all cases.

If the server and the client do not have a transport in common, or if one of them supports only IPv4 and the other only IPv6, no exchange is possible; the client may resort to using a proxy.

E.2. Shortcomings

While the mechanism above would have unified the CoAP transports under a pair of schemes, it would have rendered the use of IP literals impossible: The URI coap://[2001:db8::1] would be ambiguous as to whether CoAP-over-UDP or CoAP-over-TCP should be used. Appendix F provides a solution for this issue.

Appendix F. Literals beyond IP addresses

[This section is placed here preliminarily: After initial review in CoRE, this may be better moved into a separate document aiming for a wider IETF audience.]

F.1. Motivation for new literal-ish names

IP literals were part of URIs from the start [w3address]. Initially, they were equivalent to host names in their expressiveness, save for their inherent difference that the former can be used without a shared resolver, and the latter can be switched to a different network address.

This equivalence got lost gradually: Certificates for TLS (its precursor SSL has been available since 1995 [evossl]) have only practically been available to host names. The Host header introduced in HTTP 1.1 Section 14.23 of [RFC2616] introduced name based virtual hosting in 1999. DANE [RFC6698], which provides TLS public keys augmenting the or outside of the public key infrastructure, is only available for host names resolved through DNSSEC. SVCB records [RFC9460] introduced in 2023 allow starting newer HTTP transports without going through HTTP/1.1 first, enables load balancing, fail-over, and enable Encrypted Client Hello -- again, only for host names resolved through DNS.

This document proposes an expression for the host component of a URI that fills that gap. Note that no attempt is yet made to register service.arpa in the .ARPA Zone Management; that name is used only for the purpose of discussion.

The structure and even more the syntax used here is highly preliminary. They serves to illustrate that the desired properties can be obtained, and do not claim to be optimal. As one of many aspects, they are missing considerations for normalization and for internationalization.

F.2. Structure of service.arpa

Names under service.arpa are structured into an optional custom prefix, an ordered list of key-value component pairs, and the common suffix service.arpa.

The custom prefix can contain user defined components. The intended use is labelling, and to differentiate services provided by a single host. Any data is allowed within the structure of a URI (ABNF reg-name) and DNS name rules (63 bytes per segment). (While not ever carried by DNS, this upholds the constraints of DNS for names. That decision may be revised later, but is upheld while demonstrating that the desired properties can be obtained).

Component pairs consist of a registered component type (no precise registry is proposed at this early point) followed by encoded data. The component type "--" is special in that it concatenates the values to its left and to its right, creating component values that may exceed 63 bytes in length.

Initial component types are:

* "6": The value encodes an IPv6 address in [RFC5952] format, with the colon character (":") replaced with a dash ("-").

  It indicates an address of a host providing the service.

If any address information is present, a client SHOULD use that information to access the service.

*"4": The value encodes an IPv4 address in dotted decimal format [RFC1123], with the dot character (".") replaced with a dash ("-").

It indicates an address of a host providing the service.

*"tlsa": The data of a DNS TLSA record [RFC6698] encoded in base32 [RFC4648].

Depending on the usage, this describes TLS credentials through which the service can be authenticated.

If present, a client MUST establish a secure connection, and MUST fail the connection if the TLSA record's requirements are not met.

*"s": Service Parameters [RFC9460]). SvcbParams in base32 encoding of their wire format.

TBD: There is likely a transformation of the parameters' presentation format that is compatible with the requirements of the authority component, but this will require some more work on the syntax.

If present, a client SHOULD use these hints to establish a connection.

TBD: Encoding only the SvcParams and not priorities and targets appears to be the right thing to do for the immediate record, but does not enable load balancing and failover. Further work is required to explore how those can be expressed, and how data pertaining to the target can be expressed, possibly in a nested structure.

F.3. Syntax of service.arpa

```
name = [ custom ".-." ] *(component) "service.arpa"

custom = reg-name
component = 1*63nodot "." comptype "."
comptype = nodotnodash /  2*63nodot

; unreserved/subdelims without dot
nodot  = nodotnodash / "-"
; unreserved/subdelims without dot or dash
nodotnodash = ALPHA / DIGIT / "_" / "~" / sub-delims

; reg-name and sub-delims as in RFC3986
```

Due to [RFC3986]'s rules, all components are case insensitive and canonically lowercase.

Note that while using the IPvFuture mechanism of [RFC3986] would avoid compatibility issues around the 63 character limit and some of the character restrictions, it would not resolve the larger limitation of case insensitivity.

F.4. Processing service.arpa

A client accessing a resource under a service.arpa name does not consult DNS, but obtains information equivalent to the results of a DNS query from parsing the name.

DNS resolvers should refuse to resolve service.arpa names. (They would have all the information needed to produce sensible results, but operational aspects would need a lot of consideration; future versions of this document may revise this).

F.5. Examples

TBD: For SvcParams, the examples are inconsistent with the base32 recommendation; they serve to explore the possible alternatives.

* http://s.alpn_h2c.6.2001-db8--1.service.arpa/ -- The server is reachable on 2001:db8::1 using HTTP/2

* https://mail.-.tlsa.amaqckrkfivcukrkfivcukrkfivcukrkfivcukrkfivcukrkfivcukrk.service.arpa/ -- No address information is provided, the client needs to resort to other discovery mechanisms. Any server is eligible to serve the resource if it can present a (possibly self-signed) certificate whose public key SHA256 matches the encoded value. The "mail.-." part is provided to the server as part of the Host header, and can be used for name based virtual hosting.

*coap://s.coaptransfer_tcp_coapsecurity_edhoc.6.2001-db8--1.service.arpa/ -- The server is reachable using CoAP over TCP with EDHOC security at 2001:db8::1. (The SVCB parameters are experimental values from [I-D.lenders-core-dnr]).

Appendix G. Acknowledgements

This document heavily builds on concepts explored by Bill Silverajan and Mert Ocak in [I-D.silverajan-core-coap-protocol-negotiation], and together with Ines Robles and Klaus Hartke inside T2TRG.

[TBD: reviewers Marco Klaus]

Authors' Addresses

Christian Amsüss
Austria

Email: christian@amsuess.com

Martine Sophie Lenders
TUD Dresden University of Technology
Helmholtzstr. 10
D-01069 Dresden
Germany

Email: martine.lenders@tu-dresden.de