anima Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 8, 2021

M. Richardson
Sandelman Software Works
P. van der Stok
vanderstok consultancy
P. Kampanakis
Cisco Systems

February 04, 2021

Constrained Join Proxy for Bootstrapping Protocols
draft-ietf-anima-constrained-join-proxy-02

Abstract

This document defines a protocol to securely assign a pledge to a domain, represented by a Registrar, using an intermediary node between pledge and Registrar. This intermediary node is known as a "constrained Join Proxy".

This document extends the work of [I-D.ietf-anima-bootstrapping-keyinfra] by replacing the Circuit-proxy by a stateless/stateful constrained (CoAP) Join Proxy. It transports join traffic from the pledge to the Registrar without requiring per-client state.

Table of Contents

1. Introduction

Enrolment of new nodes into networks with enrolled nodes present is described in [I-D.ietf-anima-bootstrapping-keyinfra] ("BRSKI") and makes use of Enrolment over Secure Transport (EST) [RFC7030] with [RFC8366] vouchers to securely enroll devices. BRSKI connects new devices ("pledges") to "Registrars" via a Join Proxy.

The specified solutions use https and may be too large in terms of code space or bandwidth required for constrained devices. Constrained devices possibly part of constrained networks [RFC7228] typically implement the IPv6 over Low-Power Wireless personal Area Networks (6LoWPAN) [RFC4944] and Constrained Application Protocol (CoAP) [RFC7252].

CoAP can be run with the Datagram Transport Layer Security (DTLS) [RFC6347] as a security protocol for authenticity and confidentiality of the messages. This is known as the "coaps" scheme. A constrained version of EST, using Coap and DTLS, is described in [I-D.ietf-ace-coap-est]. The {I-D.ietf-anima-constrained-voucher} describes the BRSKI extensions to the Registrar.

DTLS is a client-server protocol relying on the underlying IP layer to perform the routing between the DTLS Client and the DTLS Server. However, the new "joining" device will not be IP routable until it is authenticated to the network. A new "joining" device can only initially use a link-local IPv6 address to communicate with a neighbour node using neighbour discovery [RFC6775] until it receives the necessary network configuration parameters. However, before the device can receive these configuration parameters, it needs to authenticate itself to the network to which it connects. IPv6 routing is necessary to establish a connection between joining device and the Registrar.

A DTLS connection is required between Pledge and Registrar.

This document specifies a new form of Join Proxy and protocol to act as intermediary between joining device and Registrar to establish a connection between joining device and Registrar.

This document is very much inspired by text published earlier in [I-D.kumar-dice-dtls-relay].
[I-D.richardson-anima-state-for-joinrouter] outlined the various options for building a join proxy.
[I-D.ietf-anima-bootstrapping-keyinfra] adopted only the Circuit Proxy method (1), leaving the other methods as future work. This document standardizes the CoAP/DTLS (method 4).

2. Terminology

The following terms are defined in [RFC8366], and are used identically as in that document: artifact, imprint, domain, Join Registrar/Coordinator (JRC), Manufacturer Authorized Signing Authority (MASA), pledge, Trust of First Use (TOFU), and Voucher.

3. Requirements Language

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.

4. Join Proxy functionality

As depicted in the Figure 1, the joining Device, or pledge (P), in an LLN mesh can be more than one hop away from the Registrar (R) and not yet authenticated into the network.

In this situation, it can only communicate one-hop to its nearest neighbour, the Join Proxy (J) using their link-local IPv6 addresses. However, the Pledge (P) needs to communicate with end-to-end security with a Registrar hosting the Registrar (R) to authenticate and get the relevant system/network parameters. If the Pledge (P) initiates a DTLS connection to the Registrar whose IP address has been pre-configured, then the packets are dropped at the Join Proxy (J) since the Pledge (P) is not yet admitted to the network or there is no IP routability to Pledge (P) for any returned messages.

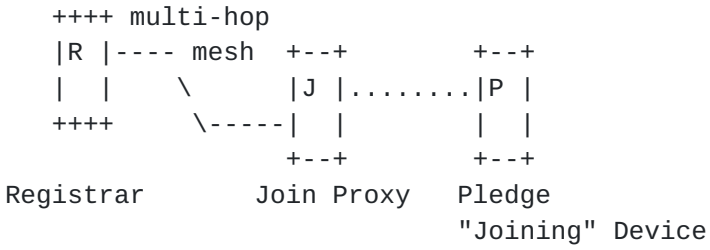


Figure 1: multi-hop enrolment.

Without routing the Pledge (P) cannot establish a secure connection to the Registrar (R) in the network assuming appropriate credentials are exchanged out-of-band, e.g. a hash of the Pledge (P)'s raw public key could be provided to the Registrar (R).

Furthermore, the Pledge (P) may be unaware of the IP address of the Registrar (R) to initiate a DTLS connection and perform authentication.

To overcome the problems with non-routability of DTLS packets and/or discovery of the destination address of the EST Server to contact, the Join Proxy is introduced. This Join Proxy functionality is configured into all authenticated devices in the network which may act as the Join Proxy for newly joining nodes. The Join Proxy allows for routing of the packets from the Pledge using IP routing to the intended Registrar.

5. Join Proxy specification

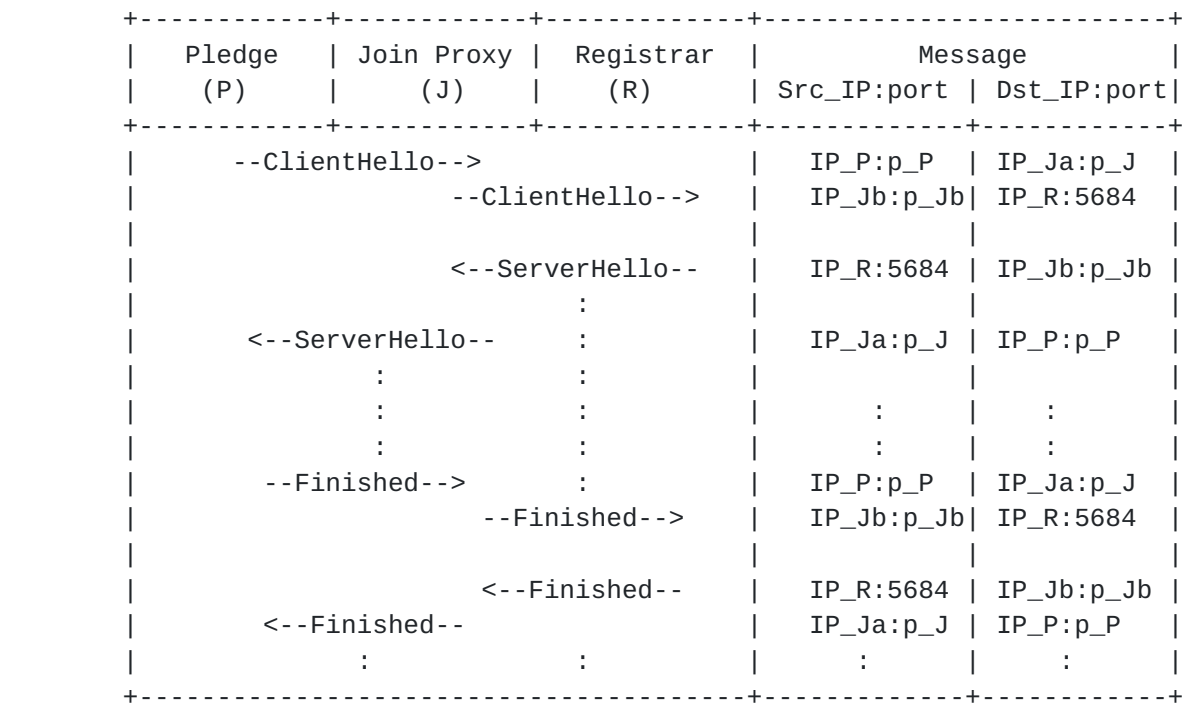
A Join Proxy can operate in two modes:

- Statefull mode
- Stateless mode

5.1. Statefull Join Proxy

In stateful mode, the joining node forwards the DTLS messages to the Registrar.

Assume that the Pledge does not know the IP address of the Registrar it needs to contact. The Join Proxy has has been enrolled via the Registrar and consequently knows the IP address and port of the Registrar. The Pledge first discovers and selects the most appropriate Join Proxy. (Discovery can be based upon [I-D.ietf-anima-bootstrapping-keyinfra] section 4.3, or via DNS-SD service discovery [RFC6763]). The Pledge initiates its request as if the Join Proxy is the intended Registrar. The Join Proxy receives the message at a discoverable "Join" port. The Join Proxy changes the IP packet (without modifying the DTLS message) by modifying both the source and destination addresses to forward the message to the intended Registrar. The Join Proxy maintains a 4-tuple array to translate the DTLS messages received from the Registrar and forward it to the EST Client. This is a form of Network Address translation, where the Join Proxy acts as a forward proxy. In Figure 2 the various steps of the message flow are shown, with 5684 being the standard coaps port:



IP_P:p_P = Link-local IP address and port of Pledge (DTLS Client)
IP_R:5684 = Global IP address and coaps port of Registrar
IP_Ja:P_J = Link-local IP address and join port of Join Proxy
IP_Jb:p_Rb = Global IP address and client port of Join proxy

Figure 2: constrained statefull joining message flow with Registrar address known to Join Proxy.

5.2. Stateless Join Proxy

The stateless Join Proxy aims to minimize the requirements on the constrained Join Proxy device. Stateless operation requires no memory in the Join Proxy device, but may also reduce the CPU impact as the device does not need to search through a state table.

If an untrusted Pledge that can only use link-local addressing wants to contact a trusted Registrar, and the Registrar is more than one hop away, it sends the DTLS message to the Join Proxy.

When a Pledge attempts a DTLS connection to the Join Proxy, it uses its link-local IP address as its IP source address. This message is transmitted one-hop to a neighbouring (Join Proxy) node. Under normal circumstances, this message would be dropped at the neighbour node since the Pledge is not yet IP routable or is not yet authenticated to send messages through the network. However, if the neighbour device has the Join Proxy functionality enabled, it routes the DTLS message to its Registrar of choice.

The Join Proxy extends this message into a new type of message called Join ProxY (JPY) message and sends it on to the Registrar.

The JPY message payload consists of two parts:

- Header (H) field: consisting of the source link-local address and port of the Pledge (P), and

- Contents (C) field: containing the original DTLS message.

On receiving the JPY message, the Registrar retrieves the two parts.

The Registrar transiently stores the Header field information. The Registrar uses the Contents field to execute the Registrar functionality. However, when the Registrar replies, it also extends its DTLS message with the header field in a JPY message and sends it back to the Join Proxy. The Registrar SHOULD NOT assume that it can decode the Header Field, it should simply repeat it when responding. The Header contains the original source link-local address and port of the pledge from the transient state stored earlier and the Contents field contains the DTLS message.

On receiving the JPY message, the Join Proxy retrieves the two parts. It uses the Header field to route the DTLS message retrieved from the Contents field to the Pledge.

In this scenario, both the Registrar and the Join Proxy use discoverable "Join" ports.

The Figure 3 depicts the message flow diagram:

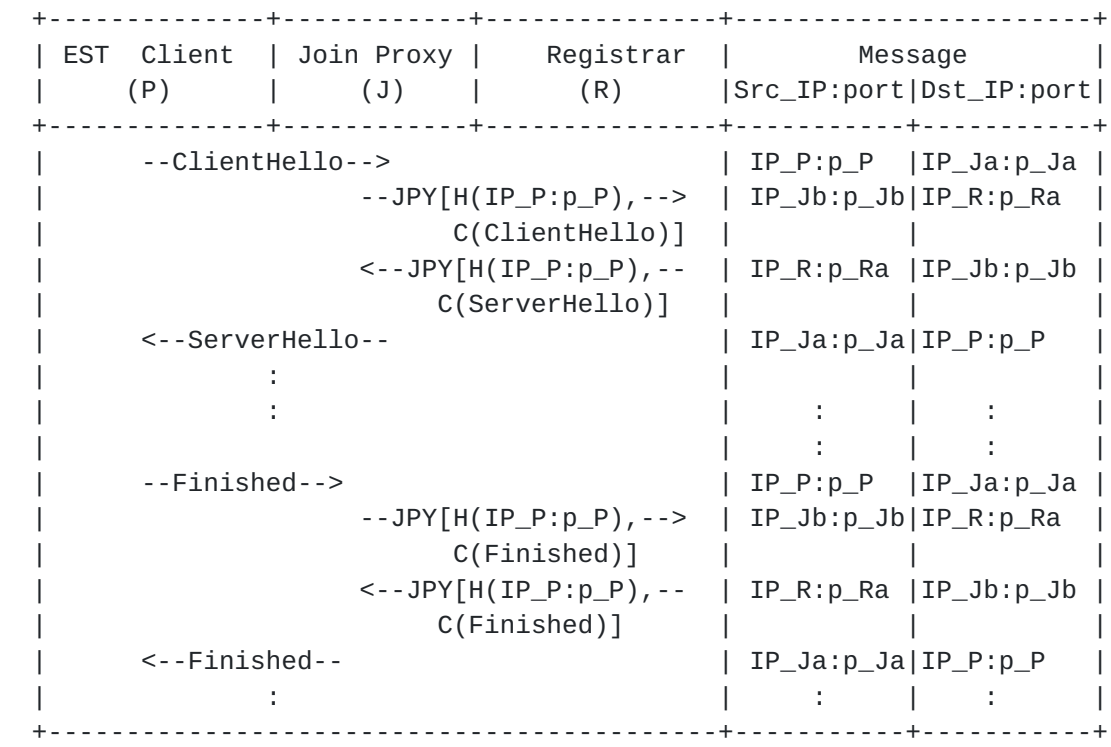


Figure 3: constrained stateless joining message flow.

5.3. Stateless Message structure

The JPY message is constructed as a payload with media-type aplication/cbor

Header and Contents fields togther are one cbor array of 5 elements:

1. header field: containing a CBOR array [RFC7049] with the pledge IPv6 Link Local address as a cbor byte string, the pledge's UDP port number as a CBOR integer, the IP address family (IPv4/IPv6) as a cbor integer, and the proxy's ifindex or other identifier for the physical port as cbor integer. The header field is not DTLS encrypted.

2. Content field: containing the DTLS encrypted payload as a CBOR byte string.

The join_proxy cannot decrypt the DTLS ecrypted payload and has no knowledge of the transported media type.

```
JPY_message =
[
   ip      : bstr,
   port    : int,
   family  : int,
   index   : int
   payload : bstr
]
```

Figure 4: CDDL representation of JPY message

The content fields are DTLS encrypted. In CBOR diagnostic notation the payload JPY[H(IP_P:p_P)], will look like:

```
[h'IP_p', p_P, family, ident, h'DTLS-content']
```

Examples are shown in Appendix A.

6. Comparison of stateless and statefull modes

The stateful and stateless mode of operation for the Join Proxy have their advantages and disadvantages. This section should enable to make a choice between the two modes based on the available device resources and network bandwidth.

Properties	Stateful mode	Stateless mode
State Information	The Join Proxy needs additional storage to maintain mapping between the address and port number of the pledge and those of the Registrar.	No information is maintained by the Join Proxy. Registrar needs to store the packet header.
Packet size	The size of the forwarded message is the same as the original message.	Size of the forwarded message is bigger than the original,it includes additional source and destination addresses.
Specification complexity	The Join Proxy needs additional functionality to maintain state information, and modify the source and destination addresses of the DTLS handshake messages	New JPY message to encapsulate DTLS message The Registrar and the Join Proxy have to understand the JPY message in order to process it.
Ports	Join Proxy needs discoverable "Join" port	Join Proxy and Registrar need discoverable "Join" ports

Figure 5: Comparison between stateful and stateless mode

7. Discovery

It is assumed that Join Proxy seamlessly provides a coaps connection between Pledge and coaps Registrar. In particular this section replaces section 4.2 of [I-D.ietf-anima-bootstrapping-keyinfra].

The discovery follows two steps:

1. The pledge is one hop away from the Registrar. The pledge discovers the link-local address of the Registrar as described in {I-D.ietf-ace-coap-est}. From then on, it follows the BRSKI process as described in {I-D.ietf-ace-coap-est}, using link-local addresses.

2. The pledge is more than one hop away from a relevant Registrar, and discovers the link-local address and join port of a Join

Proxy. The pledge then follows the BRSKI procedure using the link-local address of the Join Proxy.

3. The stateless Join Proxy discovers the join port of the Registrar

Once a pledge is enrolled, it may function as Join Proxy. The Join Proxy functions are advertised as descibed below. In principle, the Join Proxy functions are offered via a "join" port, and not the standard coaps port. Also the Registrar offer a "join" port to which the stateless join proxy sends the JPY message. The Join Proxy and Registrar MUST show the extra join port number when reponding to the .well-known/core request addressed to the standard coap/coaps port.

Three discovery cases are discussed: coap discovery, 6tisch discovery and GRASP discovery.

7.1. Pledge discovery of Registrar

The Pledge and Join Proxy are assumed to communicate via Link-Local addresses.

7.1.1. CoAP discovery

The discovery of the coaps Registrar, using coap discovery, by the Join Proxy follows section 6 of [I-D.ietf-ace-coap-est]. The extension to discover the additional port needed by the stateless proxy is described in Section 7.2.2.

7.1.2. Autonomous Network

In the context of autonomous networks, the Join Proxy uses the DULL GRASP M_FLOOD mechanism to announce itself. Section 4.1.1 of [I-D.ietf-anima-bootstrapping-keyinfra] discusses this in more detail. The Registrar announces itself using ACP instance of GRASP using M_FLOOD messages. Autonomous Network Join Proxies MUST support GRASP discovery of Registrar as decribed in section 4.3 of [I-D.ietf-anima-bootstrapping-keyinfra] .

7.1.3. 6tisch discovery

The discovery of Registrar by the pledge uses the enhanced beacons as discussed in [I-D.ietf-6tisch-enrollment-enhanced-beacon].

7.2. Pledge discovers Join Proxy

7.2.1. Autonomous Network

The pledge MUST listen for GRASP M_FLOOD [I-D.ietf-anima-grasp] announcements of the objective: "AN_Proxy". See section Section 4.1.1 [I-D.ietf-anima-bootstrapping-keyinfra] for the details of the objective.

7.2.2. CoAP discovery

In the context of a coap network without Autonomous Network support, discovery follows the standard coap policy. The Pledge can discover a Join Proxy by sending a link-local multicast message to ALL CoAP Nodes with address FF02::FD. Multiple or no nodes may respond. The handling of multiple responses and the absence of responses follow section 4 of [I-D.ietf-anima-bootstrapping-keyinfra].

The join port of the Join Proxy is discovered by sending a GET request to "/.well-known/core" including a resource type (rt) parameter with the value "brski-proxy" [RFC6690]. Upon success, the return payload will contain the join port.

The example below shows the discovery of the join port of the Join Proxy.

```
REQ: GET coap://[FF02::FD]/.well-known/core?rt=brski-proxy

RES: 2.05 Content
<coaps://[IP_address]:join-port>; rt="brski-proxy"
```

Port numbers are assumed to be the default numbers 5683 and 5684 for coap and coaps respectively (sections 12.6 and 12.7 of [RFC7252] when not shown in the response. Discoverable port numbers are usually returned for Join Proxy resources in the <href> of the payload (see section 5.1 of [I-D.ietf-ace-coap-est]).

7.3. Join Proxy discovers Registrar join port

7.3.1. CoAP discovery

The stateless Join Proxy can discover the join port of the Registrar by sending a GET request to "/.well-known/core" including a resource type (rt) parameter with the value "join-proxy" [RFC6690]. Upon success, the return payload will contain the join Port of the Registrar.

```
REQ: GET coap://[IP_address]/.well-known/core?rt=brski-proxy

RES: 2.05 Content
<coaps://[IP_address]:join-port>; rt="join-proxy"
```

The discoverable port numbers are usually returned for Join Proxy resources in the <href> of the payload (see section 5.1 of [I-D.ietf-ace-coap-est]).

8. Security Considerations

It should be noted here that the contents of the CBOR map used to convey return address information is not protected. However, the communication is between the Proxy and a known registrar are over the already secured portion of the network, so are not visible to eavesdropping systems.

All of the concerns in [I-D.ietf-anima-bootstrapping-keyinfra] section 4.1 apply. The pledge can be deceived by malicious AN_Proxy announcements. The pledge will only join a network to which it receives a valid [RFC8366] voucher.

If the proxy/Registrar was not over a secure network, then an attacker could change the cbor array, causing the pledge to send traffic to another node. If the such scenario needed to be supported, then it would be reasonable for the Proxy to encrypt the CBOR array using a locally generated symmetric key. The Registrar would not be able to examine the result, but it does not need to do so. This is a topic for future work.

9. IANA Considerations

This document needs to create a registry for key indices in the CBOR map. It should be given a name, and the amending formula should be IETF Specification.

9.1. Resource Type registry

This specification registers a new Resource Type (rt=) Link Target Attributes in the "Resource Type (rt=) Link Target Attribute Values" subregistry under the "Constrained RESTful Environments (CoRE) Parameters" registry.

rt="brski-proxy". This BRSKI resource is used to query and return the supported BRSKI port of the Join Proxy.

rt="join-proxy". This BRSKI resource is used to query and return the supported BRSKI port of the Registrar.

10. Acknowledgements

Many thanks for the comments by Brian Carpenter and Esko Dijk.

11. Contributors

Sandeep Kumar, Sye loong Keoh, and Oscar Garcia-Morchon are the co-authors of the draft-kumar-dice-dtls-relay-02. Their draft has served as a basis for this document. Much text from their draft is copied over to this draft.

12. Changelog

12.1. 01 to 02

- Discovery of Join Proxy and Registrar ports

12.2. 00 to 01

- Registrar used throughout instead of EST server
- Emphasized additional Join Proxy port for Join Proxy and Registrar
- updated discovery accordingly
- updated stateless Join Proxy JPY header
- JPY header described with CDDL
- Example simplified and corrected

12.3. 00 to 00

- copied from vanderstok-anima-constrained-join-proxy-05

13. References

13.1. Normative References

[I-D.ietf-6tisch-enrollment-enhanced-beacon]
Dujovne, D. and M. Richardson, "IEEE 802.15.4 Information Element encapsulation of 6TiSCH Join and Enrollment Information", draft-ietf-6tisch-enrollment-enhanced-beacon-14 (work in progress), February 2020.

[I-D.ietf-ace-coap-est]
Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST over secure CoAP (EST-coaps)", draft-ietf-ace-coap-est-18 (work in progress), January 2020.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-45 (work in progress), November 2020.

[I-D.ietf-anima-constrained-voucher]
Richardson, M., Stok, P., and P. Kampanakis, "Constrained Voucher Artifacts for Bootstrapping Protocols", draft-ietf-anima-constrained-voucher-09 (work in progress), November 2020.

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-15 (work in progress), July 2017.

[I-D.ietf-core-multipart-ct]
Fossati, T., Hartke, K., and C. Bormann, "Multipart Content-Format for CoAP", draft-ietf-core-multipart-ct-04 (work in progress), August 2019.

[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>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[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>.

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/info/rfc8366>.

13.2. Informative References

[I-D.kumar-dice-dtls-relay]
Kumar, S., Keoh, S., and O. Garcia-Morchon, "DTLS Relay for Constrained Environments", draft-kumar-dice-dtls-relay-02 (work in progress), October 2014.

[I-D.richardson-anima-state-for-joinrouter]
Richardson, M., "Considerations for stateful vs stateless join router in ANIMA bootstrap", draft-richardson-anima-state-for-joinrouter-03 (work in progress), September 2020.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

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

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <https://www.rfc-editor.org/info/rfc6775>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/info/rfc7030>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

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

Appendix A. Stateless Proxy payload examples

The examples show the get coaps://[192.168.1.200]:5965/est/crts to a Registrar. The header generated between Client and registrar and from registrar to client are shown in detail. The DTLS encrypted code is not shown.

The request from Join Proxy to Registrar looks like:

```
   85                                  # array(5)
      50                               # bytes(16)
         00000000000000000000FFFFC0A801C8 #
      19 BDA7                          # unsigned(48551)
      0A                               # unsigned(10)
      00                               # unsigned(0)
      58 2D                            # bytes(45)
   <cacrts DTLS encrypted request>
```

In CBOR Diagnostic:

```
    [h'00000000000000000000FFFFC0A801C8', 48551, 10, 0,
     h'<cacrts DTLS encrypted request>']
```

The response is:

```
   85                                  # array(5)
      50                               # bytes(16)
         00000000000000000000FFFFC0A801C8 #
      19 BDA7                          # unsigned(48551)
      0A                               # unsigned(10)
      00                               # unsigned(0)
   59 026A                             # bytes(618)
      <cacrts DTLS encrypted response>
```

In CBOR diagnostic:

```
    [h'00000000000000000000FFFFC0A801C8', 48551, 10, 0,
    h'<cacrts DTLS encrypted response>']
```

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Peter van der Stok
vanderstok consultancy

Email: consultancy@vanderstok.org

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com