ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: April 15, 2018

M. Pritikin
Cisco
M. Richardson
SSW
M. Behringer
Cisco
S. Bjarnason
Arbor Networks
K. Watsen
Juniper Networks

October 12, 2017

Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-08

Abstract

This document specifies automated bootstrapping of a remote secure key infrastructure (BRSKI) using vendor installed X.509 certificate, in combination with a vendor's authorizing service, both online and offline. Bootstrapping a new device can occur using a routable address and a cloud service, or using only link-local connectivity, or on limited/disconnected networks. Support for lower security models, including devices with minimal identity, is described for legacy reasons but not encouraged. Bootstrapping is complete when the cryptographic identity of the new key infrastructure is successfully deployed to the device but the established secure connection can be used to deploy a locally issued certificate to the device as well.

Table of Contents

1. Introduction

BRSKI provides a foundation to securely answer the following questions between an element of the network domain called the "Registrar" and an unconfigured and untouched device called a "Pledge":

- Registrar authenticating the Pledge: "Who is this device? What is its identity?"

- Registrar authorization the Pledge: "Is it mine? Do I want it? What are the chances it has been compromised?"

- Pledge authenticating the Registrar/Domain: "What is this domain's identity?"

- Pledge authorization the Registrar: "Should I join it?"

This document details protocols and messages to the endpoints to answer the above questions. The Registrar actions derive from Pledge identity, third party cloud service communications, and local access control lists. The Pledge actions derive from a cryptographically protected "voucher" message delivered through the Registrar but originating at a Manufacturer Authorized Signing Authority.

The syntactic details of vouchers are described in detail in [I-D.ietf-anima-voucher]. This document details automated protocol mechanisms to obtain vouchers, including the definition of a necessary 'voucher request' message that is a minor extension to the voucher format (see Section 3).

BRSKI results in the Pledge storing an X.509 root certificate sufficient for verifying the Registrar identity. In the process a TLS connection is established which can be directly used for Enrollment over Secure Transport (EST). In effect BRSKI provides an automated mechanism for the "Bootstrap Distribution of CA Certificates" described in [RFC7030] Section 4.1.1 wherein the Pledge "MUST [...]. engage a human user to authorize the CA certificate using out-of-band" information". With BRSKI the Pledge now can automate this process using the voucher. Integration with a complete EST enrollment is optional but trivial.

BRSKI is agile enough to support bootstrapping alternative key infrastructures, such as a symmetric key solutions, but no such system is described in this document.

1.1. Other Bootstrapping Approaches

To literally "pull yourself up by the bootstraps" is an impossible action. Similarly the secure establishment of a key infrastructure without external help is also an impossibility. Today it is commonly accepted that the initial connections between nodes are insecure, until key distribution is complete, or that domain-specific keying material is pre-provisioned on each new device in a costly and non-scalable manner. Existing mechanisms are known as non-secured 'Trust

on First Use' (TOFU) [RFC7435], 'resurrecting duckling' [Stajano99theresurrecting] or 'pre-staging'.

Another approach is to try and minimize user actions during bootstrapping. The enrollment protocol EST [RFC7030] details a set of non-autonomic bootstrapping methods in this vein:

- using the Implicit Trust Anchor database (not an autonomic solution because the URL must be securely distributed),
- engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),
- using a configured Explicit TA database (not an autonomic solution because the distribution of an explicit TA database is not autonomic),
- and using a Certificate-Less TLS mutual authentication method (not an autonomic solution because the distribution of symmetric key material is not autonomic).

These "touch" methods do not meet the requirements for zero-touch.

There are "call home" technologies where the Pledge first establishes a connection to a well known vendor service using a common client-server authentication model. After mutual authentication appropriate credentials to authenticate the target domain are transfered to the Pledge. This creates serveral problems and limitations:

- the pledge requires realtime connectivity to the vendor service,
- the domain identity is exposed to the vendor service (this is a privacy concern),
- the vendor is responsible for making the authorization decisions (this is a liability concern),

BRSKI addresses these issues by defining extensions to the EST protocol for the automated distribution of vouchers.

1.2. 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 [RFC2119].

The following terms are defined for clarity:

DomainID: The domain identity is the 160-bit SHA-1 hash of the BIT STRING of the subjectPublicKey of the domain trust anchor that is stored by the Domain CA. This is consistent with the Certification Authority subject key identifier (Section 4.2.1.2 [RFC5280]) of the Domain CA's self signed root certificate. (A string value bound to the Domain CA's self signed root certificate subject and issuer fields is often colloquially used as a humanized identity value but during protocol discussions the more exact term as defined here is used).

drop ship: The physical distribution of equipment containing the "factory default" configuration to a final destination. In zero-touch scenarios there is no staging or pre-configuration during drop-ship.

imprint: The process where a device obtains the cryptographic key material to identify and trust future interactions with a network. This term is taken from Konrad Lorenz's work in biology with new ducklings: during a critical period, the duckling would assume that anything that looks like a mother duck is in fact their mother. An equivalent for a device is to obtain the fingerprint of the network's root certification authority certificate. A device that imprints on an attacker suffers a similar fate to a duckling that imprints on a hungry wolf. Securely imprinting is a primary focus of this document.[imprinting]. The analogy to Lorenz's work was first noted in [Stajano99theresurrecting].

enrollment: The process where a device presents key material to a network and acquires a network specific identity. For example when a certificate signing request is presented to a certification authority and a certificate is obtained in response.

Pledge: The prospective device, which has an identity installed by a third-party (e.g., vendor, manufacturer or integrator).

Voucher A signed statement from the MASA service that indicates to a Pledge the cryptographic identity of the Registrar it should trust. There are different types of vouchers depending on how that trust asserted. Multiple voucher types are defined in [I-D.ietf-anima-voucher]

Domain: The set of entities that trust a common key infrastructure trust anchor. This includes the Proxy, Registrar, Domain Certificate Authority, Management components and any existing entity that is already a member of the domain.

Domain CA: The domain Certification Authority (CA) provides certification functionalities to the domain. At a minimum it provides certification functionalities to a Registrar and stores the trust anchor that defines the domain. Optionally, it certifies all elements.

Join Registrar (and Coordinator): A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a Join Registrar (and Coordinator) to control this process. Typically a Join Registrar is "inside" its domain. For simplicity this document often refers to this as just "Registrar". The term JRC is used in common with other bootstrap mechanisms.

Join Proxy: A domain entity that helps the pledge join the domain. A Proxy facilitates communication for devices that find themselves in an environment where they are not provided connectivity until after they are validated as members of the domain. The pledge is unaware that they are communicating with a proxy rather than directly with a Registrar.

MASA Service: A third-party Manufacturer Authorized Signing Authority (MASA) service on the global Internet. The MASA signs vouchers. It also provides a repository for audit log information of privacy protected bootstrapping events. It does not track ownership.

Ownership Tracker: An Ownership Tracker service on the global internet. The Ownership Tracker uses business processes to accurately track ownership of all devices shipped against domains that have purchased them. Although optional this component allows vendors to provide additional value in cases where their sales and distribution channels allow for accurately tracking of such ownership. Ownership tracking information is indicated in vouchers as described in [I-D.ietf-anima-voucher]

IDevID: An Initial Device Identity X.509 certificate installed by the vendor on new equipment.

TOFU: Trust on First Use. Used similarly to [RFC7435]. This is where a Pledge device makes no security decisions but rather simply trusts the first Registrar it is contacted by. This is also known as the "resurrecting duckling" model.

1.3. Scope of solution

Questions have been posed as to whether this solution is suitable in general for Internet of Things (IoT) networks. This depends on the capabilities of the devices in question. The terminology of [RFC7228] is best used to describe the boundaries.

The solution described in this document is aimed in general at non-constrained (i.e. class 2+) devices operating on a non-Challenged network. The entire solution as described here is not intended to be useable as-is by constrained devices operating on challenged networks (such as 802.15.4 LLNs).

In many target applications, the systems involved are large router platforms with multi-gigabit inter-connections, mounted in controlled access data centers. But this solution is not exclusive to the large, it is intended to scale to thousands of devices located in hostile environments, such as ISP provided CPE devices which are drop-shipped to the end user. The situation where an order is fulfilled from distributed warehouse from a common stock and shipped directly to the target location at the request of the domain owner is explicitly supported. That stock ("SKU") could be provided to a number of potential domain owners, and the eventual domain owner will not know a-priori which device will go to which location.

The bootstrapping process can take minutes to complete depending on the network infrastructure and device processing speed. The network communication itself is not optimized for speed; for privacy reasons, the discovery process allows for the Pledge to avoid announcing it's presence through broadcasting.

This protocol is not intended for low latency handoffs. In networks requiring such things, the pledge SHOULD already have been enrolled.

Specifically, there are protocol aspects described here which might result in congestion collapse or energy-exhaustion of intermediate battery powered routers in an LLN. Those types of networks SHOULD NOT use this solution. These limitations are predominately related to the large credential and key sizes required for device authentication. Defining symmetric key techniques that meet the operational requirements is out-of-scope but the underlying protocol operations (TLS handshake and signing structures) have sufficient algorithm agility to support such techniques when defined.

The imprint protocol described here could, however, be used by non-energy constrained devices joining a non-constrained network (for instance, smart light bulbs are usually mains powered, and speak 802.11). It could also be used by non-constrained devices across a

non-energy constrained, but challenged network (such as 802.15.4). The certificate contents, and the process by which the four questions above are resolved do apply to constrained devices. It is simply the actual on-the-wire imprint protocol which could be inappropriate.

This document presumes that network access control has either already occurred, is not required, or is integrated by the proxy and registrar in such a way that the device itself does not need to be aware of the details. Although the use of an X.509 Initial Device Identity is consistant with IEEE 802.1AR [IDevID], and allows for alignment with 802.1X network access control methods, its use here is for Pledge authentication rather than network access control. Integrating this protocol with network access control, perhaps as an Extensible Authentication Protocol (EAP) method (see [RFC3748]), is out-of-scope.

1.4. Leveraging the new key infrastructure / next steps

As a result of the protocol described herein the bootstrapped devices have a common trust anchor and a certificate has optionally been issued from a local PKI. This makes it possible to automatically deploy services across the domain in a secure manner.

Services which benefit from this:

- Device management.
- Routing authentication.
- Service discovery.

The major beneficiary is that it possible to use the credentials deployed by this protocol to secure the Autonomic Control Plane (ACP) ([I-D.ietf-anima-autonomic-control-plane]).

2. Architectural Overview

The logical elements of the bootstrapping framework are described in this section. Figure 1 provides a simplified overview of the components. Each component is logical and may be combined with other components as necessary.

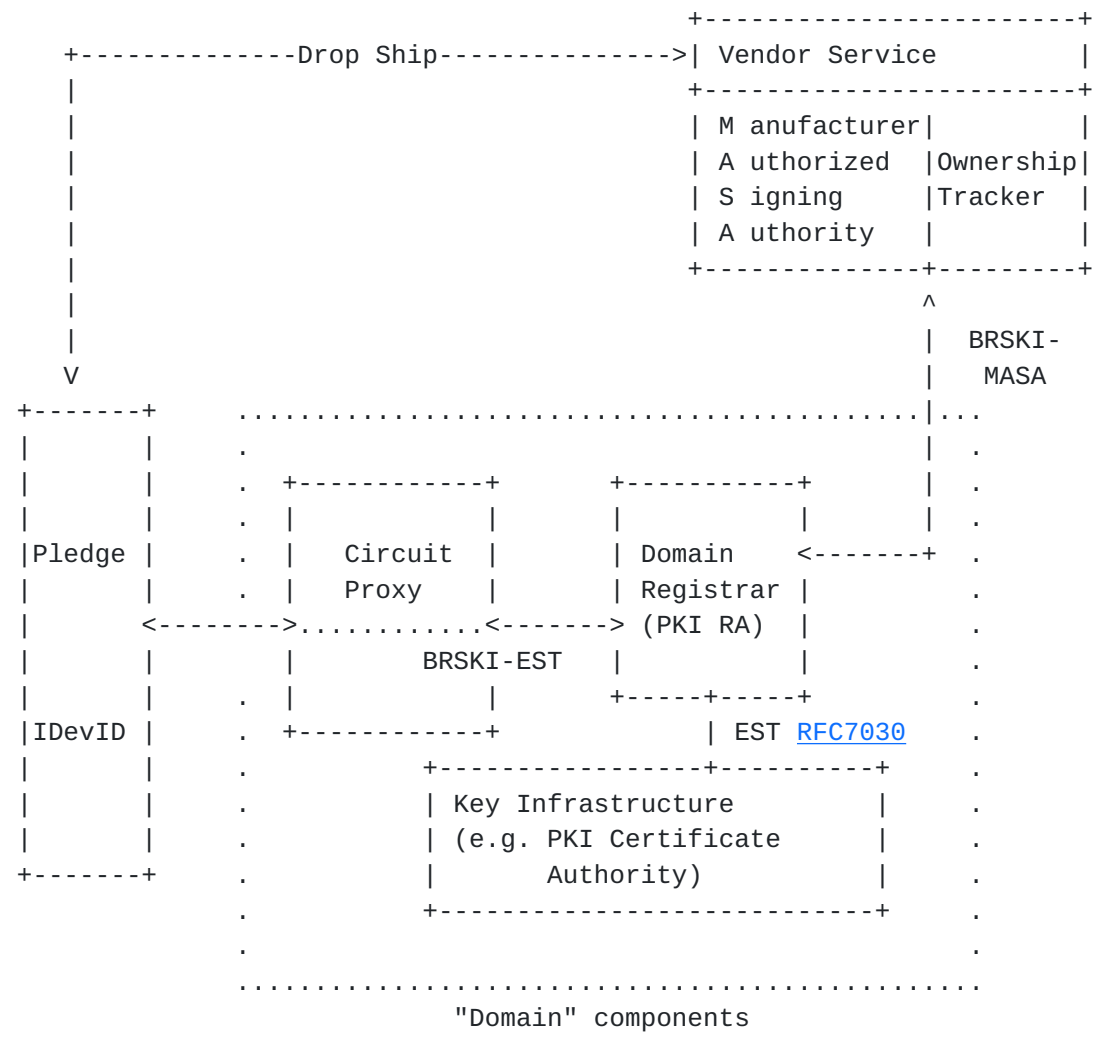


Figure 1

We assume a multi-vendor network. In such an environment there could be a Vendor Service for each vendor that supports devices following this document's specification, or an integrator could provide a generic service authorized by multiple vendors. It is unlikely that an integrator could provide Ownership Tracking services for multiple vendors due to the required sales channel integrations necessary to track ownership.

The domain is the managed network infrastructure with a Key Infrastructure the Pledge is joining. The a domain provides initial device connectivity sufficient for bootstrapping with a Circuit Proxy. The Domain Registrar authenticates the Pledge, makes authorization decisions, and distributes vouchers obtained from the Vendor Service. Optionally the Registrar also acts as a PKI Registration Authority.

2.1. Behavior of a Pledge

The pledge goes through a series of steps which are outlined here at a high level.

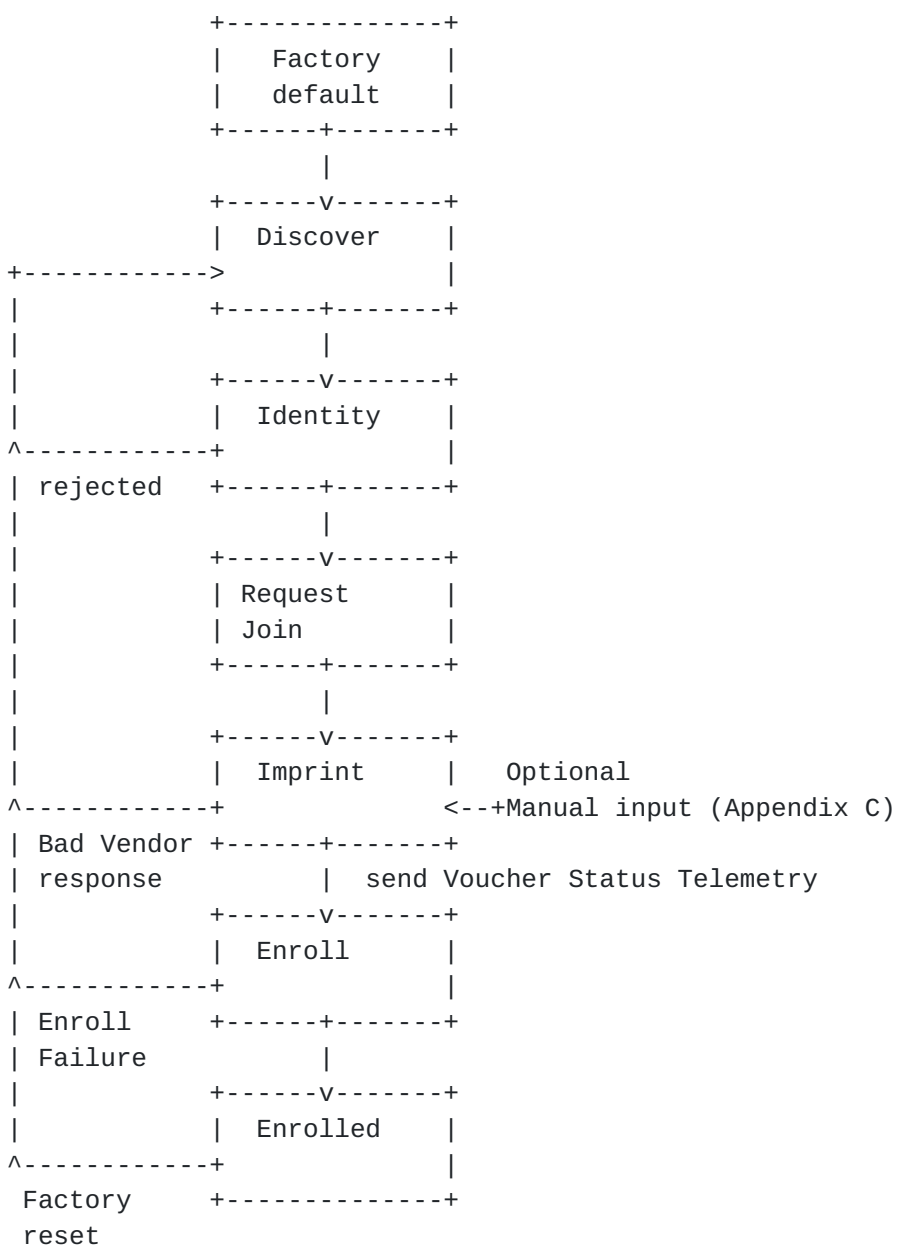


Figure 2

State descriptions for the pledge are as follows:

1. Discover a communication channel to a Registrar.

2. Identify itself. This is done by presenting an X.509 IDevID credential to the discovered Registrar (via the Proxy) in a TLS handshake. (The Registrar credentials are only provisionally accepted at this time).

3. Requests to Join the discovered Registrar. A unique nonce can be included ensuring that any responses can be associated with this particular bootstrapping attempt.

4. Imprint on the Registrar. This requires verification of the vendor service provided voucher. A voucher contains sufficient information for the Pledge to complete authentication of a Registrar. (It enables the Pledge to finish authentication of the Registrar TLS server certificate).

5. Enroll. By accepting the domain specific information from a Registrar, and by obtaining a domain certificate from a Registrar using a standard enrollment protocol, e.g. Enrollment over Secure Transport (EST) [RFC7030].

6. The Pledge is now a member of, and can be managed by, the domain and will only repeat the discovery aspects of bootstrapping if it is returned to factory default settings.

2.2. Secure Imprinting using Vouchers

A voucher is a cryptographically protected statement to the Pledge device authorizing a zero-touch imprint on the Registrar domain.

The format and cryptographic mechanism of vouchers is described in detail in [I-D.ietf-anima-voucher].

Vouchers provide a flexible mechanism to secure imprinting: the Pledge device only imprints when a voucher can be validated. At the lowest security levels the MASA server can indiscriminately issue vouchers. At the highest security levels issuance of vouchers can be integrated with complex sales channel integrations that are beyond the scope of this document. This provides the flexibility for a number of use cases via a single common protocol mechanism on the Pledge and Registrar devices that are to be widely deployed in the field. The MASA vendor services have the flexibility to leverage either the currently defined claim mechanisms or to experiment with higher or lower security levels.

Vouchers provide a signed but non-encrypted communication channel between the Pledge, the MASA, and the Registrar. The Registrar maintains control over the transport and policy decisions allowing the local security policy of the domain network to be enforced.

2.3. Initial Device Identifier

Pledge authentication and voucher request signing is via an X.509 certificate installed during the manufacturing process. This Initial Device Identifier provides a basis for authenticating the Pledge during subsequent protocol exchanges and informing the Registrar of the MASA URI. There is no requirement for a common root PKI hierarchy. Each device vendor can generate their own root certificate.

The following previously defined fields are in the X.509 IDevID certificate:

- The subject field's DN encoding MUST include the "serialNumber" attribute with the device's unique serial number.

- The subject-alt field's encoding SHOULD include a non-critical version of the RFC4108 defined HardwareModuleName.

In order to build the voucher "serial-number" field these IDevID fields need to be converted into a serial-number of "type string". The following methods is used depending on the first available IDevID certificate field (attempted in this order):

- An RFC4514 String Representation of the Distinguished Name "serialNumber" attribute.

- The HardwareModuleName hwSerialNum OCTET STRING base64 encoded.

- The RFC4514 String Representation of the Distinguished Name "common name" attribute.

The following newly defined field SHOULD be in the X.509 IDevID certificate: An X.509 non-critical certificate extension that contains a single Uniform Resource Identifier (URI) that points to an on-line Manufacturer Authorized Signing Authority. The URI is represented as described in Section 7.4 of [RFC5280].

Any Internationalized Resource Identifiers (IRIs) MUST be mapped to URIs as specified in Section 3.1 of [RFC3987] before they are placed in the certificate extension. The URI provides the authority information. The BRSKI .well-known tree is described in Section 5

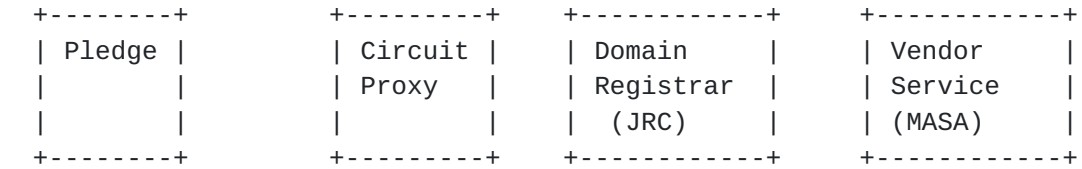
The new extension is identified as follows:

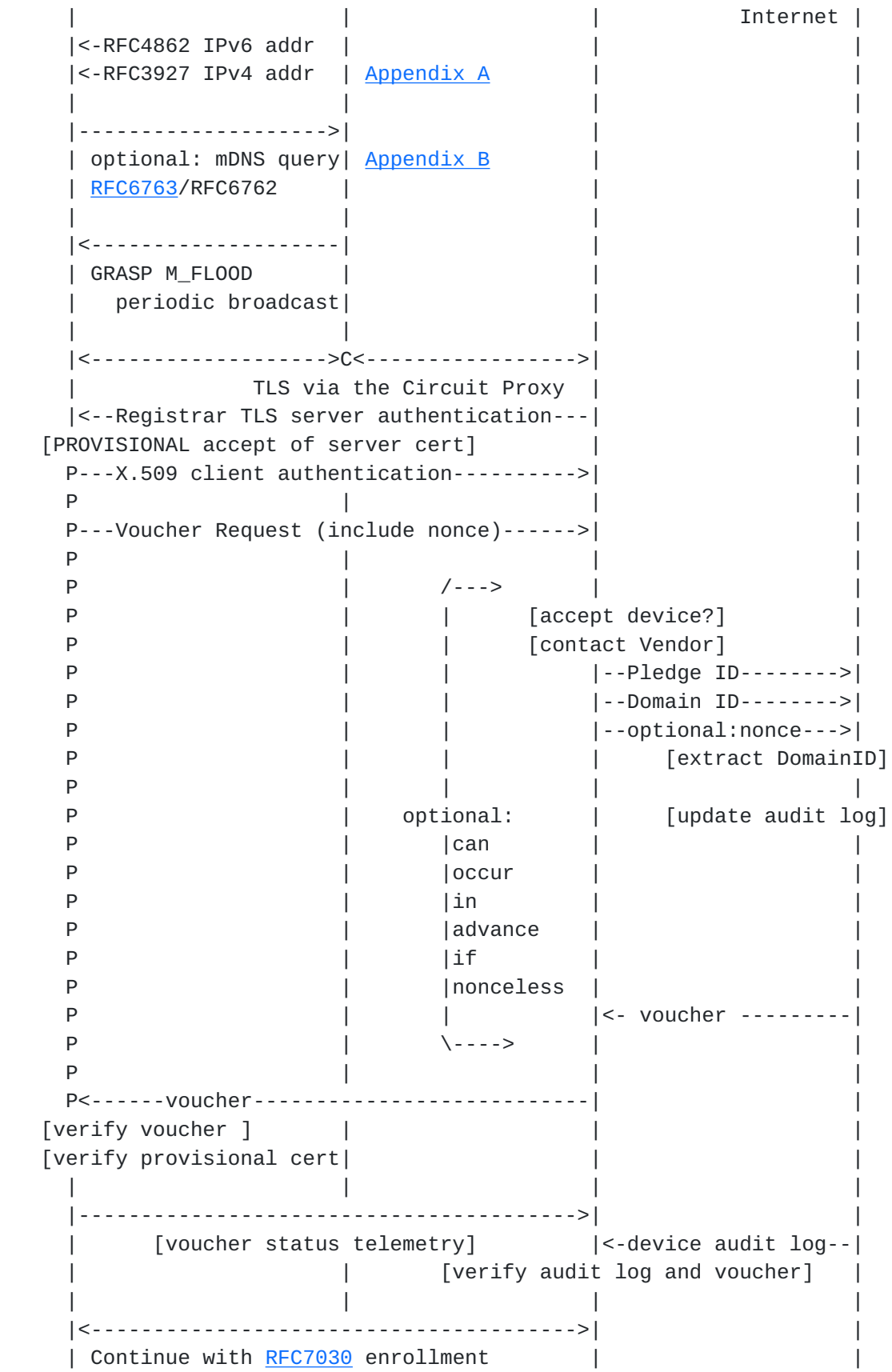
```
<CODE BEGINS>

MASAURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
internet(1) security(5) mechanisms(5) pkix(7)
id-mod(0) id-mod-MASAURLExtn2016(TBD) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS ALL --

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

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

id-pe-masa-url OBJECT IDENTIFIER ::= { id-pe TBD }

MASAURLSyntax ::= IA5String

END

<CODE ENDS>
```

The choice of id-pe is based on guidance found in Section 4.2.2 of [RFC5280], "These extensions may be used to direct applications to on-line information about the issuer or the subject". The MASA URL is precisely that: online information about the particular subject.

2.4. Protocol Flow

A representative flow is shown in Figure 3:




Internet
<-RFC4862 IPv6 addr
<-RFC3927 IPv4 addr
Appendix A
optional: mDNS query
RFC6763/RFC6762
Appendix B
GRASP M_FLOOD
periodic broadcast
C
TLS via the Circuit Proxy
<--Registrar TLS server authentication---
[PROVISIONAL accept of server cert]
P---X.509 client authentication---------->
P---Voucher Request (include nonce)------>
[accept device?]
[contact Vendor]
--Pledge ID-------->
--Domain ID-------->
--optional:nonce--->
[extract DomainID]
[update audit log]
optional:
can
occur
in
advance
if
nonceless
<- voucher ---------
P<------voucher----------
[verify voucher]
[verify provisional cert]
[voucher status telemetry]
<-device audit log--
[verify audit log and voucher]
<----------
Continue with RFC7030 enrollment

```
   | using now bidirectionally authenticated |                   |
   | TLS session.        |                   |                   |
   |                     |                   |                   |
```

Figure 3

2.4.1. Architectural component: Pledge

The Pledge is the device which is attempting to join. Until the pledge completes the enrollment process, it does has network connectivity only to the Proxy.

2.4.2. Architectural component: Circuit Proxy

The (Circuit) Proxy provides HTTPS connectivity between the pledge and the registrar. The proxy mechanism is described in Section 4, with an optional stateless mechanism described in Appendix C.

2.4.3. Architectural component: Domain Registrar

The Domain Registrar (having the formal name Join Registrar/ Coordinator (JRC)), operates as a CMC Registrar, terminating the EST and BRSKI connections. The Registrar is manually configured or distributed with a list of trust anchors necessary to authenticate any Pledge device expected on the network. The Registrar communicates with the Vendor supplied MASA to establish ownership.

2.4.4. Architectural component: Vendor Service

The Vendor Service provides two logically seperate functions: the Manufacturer Authorized Signing Authority (MASA), and an ownership tracking/auditing function.

2.5. Lack of realtime clock

Many devices when bootstrapping do not have knowledge of the current time. Mechanisms like Network Time Protocols can not be secured until bootstrapping is complete. Therefore bootstrapping is defined in a method that does not require knowledge of the current time.

Unfortunately there are moments during bootstrapping when certificates are verified, such as during the TLS handshake, where validity periods are confirmed. This paradoxical "catch-22" is resolved by the Pledge maintaining a concept of the current "window" of presumed time validity that is continually refined throughout the bootstrapping process as follows:

- Initially the Pledge does not know the current time.

- During Pledge authentiation by the Registrar a realtime clock can be used by the Registrar. This bullet expands on a closely related issue regarding Pledge lifetimes. RFC5280 indicates that long lived Pledge certifiates "SHOULD be assigned the GeneralizedTime value of 99991231235959Z" [RFC7030] so the Registrar MUST support such lifetimes and SHOULD support ignoring Pledge lifetimes if they did not follow the RFC5280 recommendations.

- The Pledge authenticates the voucher presented to it. During this authentication the Pledge ignores certificate lifetimes (by necessity because it does not have a realtime clock).

- If the voucher contains a nonce then the Pledge MUST confirm the nonce matches the original voucher request. This ensures the voucher is fresh. See / (Section 5.2).

- Once the voucher is accepted the validity period of the pinned-domain-cert in the voucher now serves as a valid time window. Any subsequent certificate validity periods checked during RFC5280 path validation MUST occur within this window.

- When accepting an enrollment certificate the validity period within the new certificate is assumed to be valid by the Pledge. The Pledge is now willing to use this credential for client authentication.

2.6. Cloud Registrar

The Pledge MAY contact a well known URI of a cloud Registrar if a local Registrar can not be discovered or if the Pledge's target use cases do not include a local Registrar.

If the Pledge uses a well known URI for contacting a cloud Registrar an Implicit Trust Anchor database (see [RFC7030]) MUST be used to authenticate service as described in RFC6125. This is consistent with the human user configuration of an EST server URI in [RFC7030] which also depends on RFC6125.

2.7. Determining the MASA to contact

The registrar needs to be able to contact a MASA that is trusted by the Pledge in order to obtain vouchers. There are three mechanisms described:

The device's Initial Device Identifier will normally contain the MASA URL as detailed in Section 2.3. This is the RECOMMENDED mechanism.

If the Registrar is integrated with [I-D.ietf-opsawg-mud] and the Pledge IDevID contains the id-pe-mud-url then the Registrar MAY attempt to obtain the MASA URL from the MUD file. The MUD file extension for the MASA URL is defined in Appendix D.

It can be operationally difficult to ensure the necessary X.509 extensions are in the Pledge's' IDevID due to the difficulty of aligning current Pledge manufacturing with software releases and development. As a final fallback the Registrar MAY be manually configured or distributed with a MASA URL for each vendor. Note that the Registrar can only select the configured MASA URL based on the trust anchor -- so vendors can only leverage this approach if they ensure a single MASA URL works for all Pledge's associated with each trust anchor.

3. Voucher Request artifact

The voucher request is how an entity requests a voucher. The Pledge forms a voucher request and submits it to the Registrar. The Registrar in turn submits a voucher request to the MASA server. A voucher request is a voucher structure with an additional "prior-signed-voucher-request" "leaf to support forwarding the Pledge's initial voucher request.

Unless otherwise signaled (outside the voucher artifact), the signing structure is as defined for vouchers, see [I-D.ietf-anima-voucher].

3.1. Tree Diagram

The following tree diagram illustrates a high-level view of a voucher request document. The notation used in this diagram is described in [I-D.ietf-anima-voucher]. Each node in the diagram is fully described by the YANG module in Section 3.3. Please review the YANG module for a detailed description of the voucher request format.

```
module: ietf-voucher-request
  groupings:
  voucher-request-grouping
      +---- voucher
         +---- created-on?                     yang:date-and-time
         +---- expires-on?                     yang:date-and-time
         +---- assertion                       enumeration
         +---- serial-number                   string
         +---- idevid-issuer?                  binary
         +---- pinned-domain-cert?             binary
         +---- domain-cert-revocation-checks?  boolean
         +---- nonce?                          binary
         +---- last-renewal-date?              yang:date-and-time
         +---- prior-signed-voucher-request?   binary
         +---- proximity-registrar-cert?       binary
```

3.2. Examples

This section provides voucher examples for illustration purposes. That these examples conform to the encoding rules defined in [RFC7951].

Example (1) The following example illustrates a Pledge generated voucher-request. The assertion leaf is indicated as 'proximity' and the Registrar's TLS server certificate is included in the 'pinned-domain-cert' leaf. See Section 5.2.

```
{
    "ietf-voucher-request:voucher": {
        "nonce": "62a2e7693d82fcda2624de58fb6722e5",
        "created-on": "2017-01-01T00:00:00.000Z",
        "assertion": "proximity",
        "proximity-registrar-cert": "base64encodedvalue=="
    }
}
```

Example (2) The following example illustrates a Registrar generated voucher-request. The 'prior-signed-voucher-request' leaf is populated with the Pledge's voucher request (such as the prior example). See Section 5.4.

```
{
    "ietf-voucher-request:voucher": {
        "nonce": "62a2e7693d82fcda2624de58fb6722e5",
        "created-on": "2017-01-01T00:00:02.000Z",
        "assertion": "proximity",
        "idevid-issuer": "base64encodedvalue=="
        "serial-number": "JADA123456789"
        "prior-signed-voucher": "base64encodedvalue=="
    }
}
```

Example (3) The following example illustrates a Registrar generated voucher-request. The 'prior-signed-voucher-request' leaf is not populated with the Pledge's voucher request nor is the nonce leaf. This form might be used by a Registrar requesting a voucher when the Pledge is offline or when the Registrar expects to be offline during deployment. See Section 5.4.

```
{
    "ietf-voucher-request:voucher": {
        "created-on": "2017-01-01T00:00:02.000Z",
        "assertion": "TBD",
        "idevid-issuer": "base64encodedvalue=="
        "serial-number": "JADA123456789"
    }
}
```

Example (4) The following example illustrates a Registrar generated voucher-request. The 'prior-signed-voucher-request' leaf is not populated with the Pledge's voucher request because the Pledge did not sign it's own request. This form might be used when more constrained Pledges are being deployed. The nonce is populated from the Pledge's request. See Section 5.4.

```
{
    "ietf-voucher-request:voucher": {
        "nonce": "62a2e7693d82fcda2624de58fb6722e5",
        "created-on": "2017-01-01T00:00:02.000Z",
        "assertion": "proximity",
        "idevid-issuer": "base64encodedvalue=="
        "serial-number": "JADA123456789"
    }
}
```

3.3. YANG Module

Following is a YANG [RFC7950] module formally extending the [I-D.ietf-anima-voucher] voucher into the voucher request.

```
<CODE BEGINS> file "ietf-voucher-request@2017-10-13.yang"
module ietf-voucher-request {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-voucher-request";
  prefix "vch";

  import ietf-restconf {
    prefix rc;
    description
      "This import statement is only present to access
       the yang-data extension defined in RFC 8040.";
    reference "RFC 8040: RESTCONF Protocol";
  }

  import ietf-voucher {
    prefix v;
    description
      "FIXME";
    reference "RFC ????: Voucher Profile for Bootstrapping Protocols";
  }

  organization
   "IETF ANIMA Working Group";

  contact
   "WG Web:   <http://tools.ietf.org/wg/anima/>
    WG List:  <mailto:anima@ietf.org>
    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>
    Author:   Max Pritikin
              <mailto:pritikin@cisco.com>
    Author:   Michael Richardson
              <mailto:mcr+ietf@sandelman.ca>
    Author:   Toerless Eckert
              <mailto:tte+ietf@cs.fau.de>";

  description
   "This module... FIXME

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT',
    'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in
```

```
    the module text are to be interpreted as described in RFC 2119.

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

    Redistribution and use in source and binary forms, with or without
    modification, is permitted pursuant to, and subject to the license
    terms contained in, the Simplified BSD License set forth in Section
    4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see the RFC
    itself for full legal notices.";

  revision "2017-10-13" {
    description
     "Initial version";
    reference
     "RFC XXXX: Voucher Profile for Bootstrapping Protocols";
  }

  // Top-level statement
  rc:yang-data voucher-request-artifact {
    uses voucher-request-grouping;
  }

  // Grouping defined for future usage
  grouping voucher-request-grouping {
    description
      "Grouping to allow reuse/extensions in future work.";

    uses v:voucher-artifact-grouping {
      refine "voucher/created-on" {
        mandatory false;
      }

      refine "voucher/pinned-domain-cert" {
        mandatory false;
      }

      augment "voucher"  {
        description
          "Adds leaf nodes appropriate for requesting vouchers.";

        leaf prior-signed-voucher-request {
          type binary;
          description
            "If it is necessary to change a voucher, or re-sign and
```

```
            forward a voucher that was previously provided along a
            protocol path, then the previously signed voucher SHOULD be
            included in this field.

            For example, a pledge might sign a proximity voucher, which
            an intermediate registrar then re-signs to make its own
            proximity assertion.  This is a simple mechanism for a
            chain of trusted parties to change a voucher, while
            maintaining the prior signature information.

            The pledge MUST ignore all prior voucher information when
            accepting a voucher for imprinting. Other parties MAY
            examine the prior signed voucher information for the
            purposes of policy decisions. For example this information
            could be useful to a MASA to determine that both pledge and
            registrar agree on proximity assertions. The MASA SHOULD
            remove all prior-signed-voucher information when signing
            a voucher for imprinting so as to minimize the final
            voucher size.";
        }

        leaf proximity-registrar-cert {
          type binary;
          description
            "An X.509 v3 certificate structure as specified by RFC 5280,
             Section 4 encoded using the ASN.1 distinguished encoding
             rules (DER), as specified in ITU-T X.690.

             The first certificate in the Registrar TLS server
             certificate_list sequence  (see [RFC5246]) presented by
             the Registrar to the Pledge. This MUST be populated in a
             Pledge's voucher request if the proximity assertion is
             populated.";
        }
      }
    }
  }

}

<CODE ENDS>
```

4. Proxy details

The role of the Proxy is to facilitate communications. The Proxy forwards packets between the Pledge and a Registrar that has been configured on the Proxy.

The Proxy does not terminate the TLS handshake: it passes streams of bytes onward without examination.

A proxy MAY assume TLS framing for auditing purposes, but MUST NOT assume any TLS version.

A Proxy is always assumed even if it is directly integrated into a Registrar. (In a completely autonomic network, the Registrar MUST provide proxy functionality so that it can be discovered, and the network can grow concentrically around the Registrar)

As a result of the Proxy Discovery process in section Section 4.1.1, the port number exposed by the proxy does not need to be well known, or require an IANA allocation.

If the Proxy joins an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]) it SHOULD use Autonomic Control Plane secured GRASP ([I-D.ietf-anima-grasp]) to discovery the Registrar address and port. As part of the discovery process, the proxy mechanism (Circuit Proxy vs IPIP encapsulation) is agreed to between the Registrar and Join Proxy.

For the IPIP encapsulation methods (described in Appendix C), the port announced by the Proxy SHOULD be the same as on the registrar in order for the proxy to remain stateless.

In order to permit the proxy functionality to be implemented on the maximum variety of devices the chosen mechanism SHOULD use the minimum amount of state on the proxy device. While many devices in the ANIMA target space will be rather large routers, the proxy function is likely to be implemented in the control plane CPU of such a device, with available capabilities for the proxy function similar to many class 2 IoT devices.

The document [I-D.richardson-anima-state-for-joinrouter] provides a more extensive analysis and background of the alternative proxy methods.

4.1. Pledge discovery of Proxy

The result of discovery is a logical communication with a Registrar, through a Proxy. The Proxy is transparent to the Pledge but is always assumed to exist.

To discover the Proxy the Pledge performs the following actions:

1. MUST: Obtains a local address using IPv6 methods as described in [RFC4862] IPv6 Stateless Address AutoConfiguration. Use of

[RFC4941] temporary addresses is encouraged. A new temporary address SHOULD be allocated whenever the discovery process is forced to restart due to failures. Pledges will generally prefer use of IPv6 Link-Local addresses, and discovery of Proxy will be by Link-Local mechanisms. IPv4 methods are described in Appendix A

2. MUST: Listen for GRASP M_FLOOD ([I-D.ietf-anima-grasp]) announcements of the objective: "AN_Proxy". See section Section 4.1.1 for the details of the objective. The Pledge may listen concurrently for other sources of information, see Appendix B.

Once a proxy is discovered the Pledge communicates with a Registrar through the proxy using the bootstrapping protocol defined in Section 5.

Each discovery method attempted SHOULD exponentially back-off attempts (to a maximum of one hour) to avoid overloading the network infrastructure with discovery. The back-off timer for each method MUST be independent of other methods.

Methods SHOULD be run in parallel to avoid head of queue problems wherein an attacker running a fake proxy or registrar can operate protocol actions intentionally slowly.

Once a connection to a Registrar is established (e.g. establishment of a TLS session key) there are expectations of more timely responses, see Section 5.2.

Once all discovered services are attempted the device SHOULD return to listening for GRASP M_FLOOD. It should periodically retry the vendor specific mechanisms. The Pledge MAY prioritize selection order as appropriate for the anticipated environment.

4.1.1. Proxy Grasp announcements

A proxy uses the GRASP M_FLOOD mechanism to announce itself. The pledge SHOULD listen for messages of these form. This announcement can be within the same message as the ACP announcement detailed in [I-D.ietf-anima-autonomic-control-plane].

```
proxy-objective = ["AN_Proxy", [ O_IPv6_LOCATOR, ipv6-address,
transport-proto, port-number ] ]

ipv6-address       - the v6 LL of the proxy
transport-proto    - 6, for TCP 17 for UDP
port-number        - the TCP or UDP port number to find the proxy
```

Figure 5

4.2. CoAP connection to Registrar

The use of CoAP to connect from Pledge to Registrar is out of scope for this document, and may be described in future work.

4.3. HTTPS proxy connection to Registrar

The proxy SHOULD also provide one of: an IPIP encapsulation of HTTP traffic to the registrar, or a TCP circuit proxy that connects the Pledge to a Registrar.

When the Proxy provides a circuit proxy to a Registrar the Registrar MUST accept HTTPS connections.

4.4. Proxy discovery of Registrar

The Registrar SHOULD announce itself so that proxies can find it and determine what kind of connections can be terminated.

When the Registrar joins an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]) it MUST respond to GRASP ([I-D.ietf-anima-grasp]) M_NEG_SYN message.

The registrar responds to discovery messages from the proxy (or GRASP caches between them) as follows: (XXX changed from M_DISCOVERY)

```
objective         = ["AN_registrar", F_DISC, 255 ]
discovery-message = [M_NEG_SYN, session-id, initiator, objective]
```

Figure 6: Registrar Discovery

The response from the registrar (or cache) will be a M_RESPONSE with the following parameters:

```
response-message = [M_RESPONSE, session-id, initiator, ttl,
(+locator-option // divert-option), ?objective)]
initiator = ACP address of Registrar
locator1  = [O_IPv6_LOCATOR, fd45:1345::6789, 6,  443]
locator2  = [O_IPv6_LOCATOR, fd45:1345::6789, 17, 5683]
locator3  = [O_IPv6_LOCATOR, fe80::1234, 41, nil]
```

Figure 7: Registrar Response

The set of locators is to be interpreted as follows. A protocol of 6 indicates that TCP proxying on the indicated port is desired. A protocol of 17 indicates that UDP proxying on the indicated port is desired. In each case, the traffic SHOULD be proxied to the same port at the ULA address provided.

A protocol of 41 indicates that packets may be IPIP proxy'ed. In the case of that IPIP proxying is used, then the provided link-local address MUST be advertised on the local link using proxy neighbour discovery. The Join Proxy MAY limit forwarded traffic to the protocol (6 and 17) and port numbers indicated by locator1 and locator2. The address to which the IPIP traffic should be sent is the initiator address (an ACP address of the Registrar), not the address given in the locator.

Registrars MUST accept TCP / UDP traffic on the ports given at the ACP address of the Registrar. If the Registrar supports IPIP tunnelling, it MUST also accept traffic encapsulated with IPIP.

Registrars MUST accept HTTPS/EST traffic on the TCP ports indicated. Registrars MAY accept DTLS/CoAP/EST traffic on the UDP in addition to TCP traffic.

5. Protocol Details

The Pledge MUST initiate BRSKI after boot if it is unconfigured. The Pledge MUST NOT automatically initiate BRSKI if it has been configured or is in the process of being configured.

BRSKI is described as extensions to EST [RFC7030] to reduce the number of TLS connections and crypto operations required on the Pledge. The Registrar implements the BRSKI REST interface within the same .well-known URI tree as the existing EST URIs as described in EST [RFC7030] section 3.2.2. The communication channel between the Pledge and the Registrar is referred to as "BRSKI-EST" (see Figure 1).

The communication channel between the Registrar and MASA is similarly described as extensions to EST within the same ./well-known tree.

For clarity this channel is referred to as "BRSKI-MASA". (See Figure 1).

MASA URI is "https:// authority "./well-known/est".

BRSKI uses EST message formats for existing operations, uses JSON [RFC7159] for all new operations defined here, and voucher formats.

While EST section 3.2 does not insist upon use of HTTP 1.1 persistent connections, BRSKI-EST connections SHOULD use persistent connections. The intention of this guidance is to ensure the provisional TLS authentication occurs only once and is properly managed.

Summarized automation extensions for the BRSKI-EST flow are:

- The Pledge provisionally accepts the Registrar certificate during the TLS handshake as detailed in Section 5.1.

- If the Registrar responds with a redirection to other web origins the Pledge MUST follow only a single redirection. (EST supports redirection but does not allow redirections to other web origins without user input).

- The Registar MAY respond with an HTTP 202 ("the request has been accepted for processing, but the processing has not been completed") as described in EST [RFC7030] section 4.2.3 wherein the client "MUST wait at least the specified 'retry-after' time before repeating the same request". The Pledge is RECOMMENDED to provide local feed (blinked LED etc) during this wait cycle if mechanisms for this are available. To prevent an attacker Registrar from significantly delaying bootstrapping the Pledge MUST limit the 'retry-after' time to 60 seconds. To avoid blocking on a single erroneous Registrar the Pledge MUST drop the connection after 5 seconds in which there has been no progress on the TCP connection. It should proceed to other discovered Registrars if there are any. If there were no other Registrars discovered, the pledge MAY continue to wait, as long as it is concurrently listening for new proxy announcements.

- Ideally the Pledge could keep track of the appropriate retry-after value for any number of outstanding Registrars but this would involve a large state table on the Pledge. Instead the pledge MAY ignore the exact retry-after value in favor of a single hard coded value that takes effect between discovery ([[ProxyDiscovery]]) attempts. A Registrar that is unable to complete the transaction the first time due to timing reasons will have future chances.

- The Pledge requests and validates a voucher using the new REST calls described below.

- If necessary the Pledge calls the EST defined /cacerts method to obtain the domain owners' CA certificate. The pinned-domain-certificate element from the voucher should validate this certificate, or be identical to it.

- The Pledge completes authentication of the server certificate as detailed in Section 5.5.1. This moves the BRSKI-EST TLS connection out of the provisional state. Optionally, the BRSKI-EST TLS connection can now be used for EST enrollment.

The extensions for a Registrar (equivalent to EST server) are:

- Client authentication is automated using Initial Device Identity (IDevID) as per the EST certificate based client authentication. The subject field's DN encoding MUST include the "serialNumber" attribute with the device's unique serial number. In the language of RFC6125 this provides for a SERIALNUM-ID category of identifier that can be included in a certificate and therefore that can also be used for matching purposes. The SERIALNUM-ID whitelist is collated according to vendor trust anchor since serial numbers are not globally unique.

- The Registrar requests and validates the Voucher from the vendor authorized MASA service.

- The Registrar forwards the Voucher to the Pledge when requested.

- The Registar performs log verifications in addition to local authorization checks before accepting optional Pledge device enrollment requests.

5.1. BRSKI-EST TLS establishment details

The Pledge establishes the TLS connection with the Registrar through the circuit proxy (see Section 4) but the TLS handshake is with the Registar. The BRSKI-EST Pledge is the TLS client and the BRSKI-EST Registrar is the TLS server. All security associations established are between the Pledge and the Registrar regardless of proxy operations.

Establishment of the BRSKI-EST TLS connection is as specified in EST [RFC7030] section 4.1.1 "Bootstrap Distribution of CA Certificates" [RFC7030] wherein the client is authenticated with the IDevID certificate, and the EST server (the Registrar) is provisionally authenticated with a unverified server certificate.

The Pledge maintains a security paranoia concerning the provisional state, and all data recieved, until a voucher is received and verified as specified in Section 5.5.1

5.2. Pledge Requests Voucher from the Registrar

When the Pledge bootstraps it makes a request for a Voucher from a Registrar.

This is done with an HTTPS POST using the operation path value of "/requestvoucher".

The request media types are:

application/pkcs7-mime; smime-type=voucher-request The request is a "YANG-defined JSON document that has been signed using a PKCS#7 structure" as described in Section 3 using the JSON encoding described in [RFC7951]. The Pledge SHOULD sign the request using the Section 2.3 credential.

application/json The request is the "YANG-defined JSON document" as described in Section 3 with exception that it is not within a PKCS#7 structure. It is protected only by the TLS client authentication. This reduces the cryptographic requirements on the Pledge.

For simplicity the term 'voucher request' is used to refer to either of these media types. Registrar impementations SHOULD anticipate future media types but of course will simply fail the request if those types are not yet known.

The Pledge populates the voucher request fields as follows:

created-on: Pledges that have a realtime clock are RECOMMENDED to populate this field. This provides additional information to the MASA.

nonce: The voucher request MUST contain a cryptographically strong random or pseudo-random number nonce. Doing so ensures Section 2.5 functionality. The nonce MUST NOT be reused for bootstrapping attempts.

assertion: The voucher request MAY contain an assertion of "proximity".

proximity-registrar-cert: In a Pledge voucher request this is the first certificate in the TLS server 'certificate_list' sequence (see [RFC5246]) presented by the Registrar to the Pledge. This

MUST be populated in a Pledge's voucher request if the "proximity" assertion is populated.

All other fields MAY be omitted in the voucher request.

An example JSON payload of a voucher request from a Pledge is in Section 3.2 Example 1.

The Registrar validates the client identity as described in EST [RFC7030] section 3.3.2. If the request is signed the Registrar confirms the 'proximity' asserion and associated 'proximity-registrar-cert' are correct. The registrar performs authorization as detailed in [[EDNOTE: UNRESOLVED. See Appendix D "Pledge Authorization"]]. If these validations fail the Registrar SHOULD respond with an appropriate HTTP error code.

If authorization is successful the Registrar obtains a voucher from the MASA service (see Section 5.4) and returns that MASA signed voucher to the pledge as described in Section 5.5.

5.3. BRSKI-MASA TLS establishment details

The BRSKI-MASA TLS connection is a 'normal' TLS connection appropriate for HTTPS REST interfaces. The Registrar initiates the connection and uses the MASA URL obtained as described in Section 2.7 for RFC6125 authentication of the MASA server.

The primary method of Registrar "authentication" by the MASA is detailed in Section 5.4. As detailed in Section 8 the MASA might find it necessary to request additional Registrar authentication. Registrars MUST be prepared to support TLS client certificate authentication and HTTP Basic or Digest authentication as described in RFC7030 for EST clients. Implementors are advised that contacting the MASA is to establish a secured REST connection with a web service and that there are a number of authentication models being explored within the industry. Registrars are RECOMMENDED to fail gracefully and generate useful administrative notifications or logs in the advent of unexpected HTTP 401 (Unauthorized) responses from the MASA.

5.4. Registrar Requests Voucher from MASA

When a Registrar receives a voucher request from a Pledge it in turn requests a voucher from the MASA service. For simplicity this is defined as an optional EST message between a Registrar and an EST server running on the MASA service although the Registrar is not required to make use of any other EST functionality when communicating with the MASA service. (The MASA service MUST properly

reject any EST functionality requests it does not wish to service; a requirement that holds for any REST interface).

This is done with an HTTP POST using the operation path value of "/requestvoucher".

The request media type is:

application/pkcs7-mime; smime-type=voucher-request The request is a "YANG-defined JSON document that has been signed using a PKCS#7 structure" as described in [I-D.ietf-anima-voucher] using the JSON encoding described in [RFC7951]. The Registrar MUST sign the request. The entire Registrar certificate chain, up to and including the Domain CA, MUST be included in the PKCS#7 structure.

For simplicity the term 'voucher request' is used. MASA impementations SHOULD anticipate future media types but of course will simply fail the request if those types are not yet known.

The Registrar populates the voucher request fields as follows:

created-on: Registrars are RECOMMENDED to populate this field. This provides additional information to the MASA.

nonce: The optional nonce value from the Pledge request if desired (see below).

serial-number: The serial number of the Pledge the Registrar would like a voucher for.

idevid-issuer: The idevid-issuer value from the pledge certificate is included to ensure a statistically unique identity. The Pledge's serial number is extracted from the X.509 IDevID. See Section 2.3.

prior-signed-voucher: If the Pledge provided a signed voucher request then it SHOULD be included in the voucher request built by the Registrar. (NOTE: this is the Pledge's complete voucher request, inclusive of the 'assertion', 'proximity-registrar-cert', etc wrapped by the pledge's original PKCS#7 signature).

A Registrar MAY exclude the nonce from the voucher request it submits to the MASA. Doing so allows the Registrar to request a Voucher when the Pledge is offline, or when the Registrar is expected to be offline when the Pledge is being deployed. These use cases require the Registrar to learn the appropriate IDevID SerialNumber field from the physical device labeling or from the sales channel (out-of-scope of this document). If a nonce is not provided the MASA server MUST

authenticate the Registrar as described in EST [RFC7030] section 3.3.2 to reduce the risk of DDoS attacks and to provide an authenticated identity as an input to sales channel integration and authorizations (also out-of-scope of this document).

All other fields MAY be omitted in the voucher request.

Example JSON payloads of voucher requests from a Registrar are in Section 3.2 Example 2 through 4.

The MASA verifies that the voucher request is internally consistent but does not authenticate the registrar certificate since the registrar is not know to the MASA server in advance. The MASA validation checks before issuing a voucher are as follows:

Renew for expired voucher: As described in [I-D.ietf-anima-voucher] vouchers are normally short lived to avoid revocation issues. If the request is for a previous (expired) voucher using the same Registrar (as determined by the Registrar pinned-domain-cert) and the MASA has not been informed that the claim is invalid then the request for a renewed voucher SHOULD be automatically authorized.

Voucher signature consistency: The MASA MUST verify that the voucher request is signed by a Registrar. This is confirmed by verifying that the id-kp-cmcRA extended key usage extension field (as detailed in EST RFC7030 section 3.6.1) exists in the certificate of the entity that signed the voucher request. This verification is only a consistency check that the unauthenticated domain CA intended this to be a Registrar. Performing this check provides value to domain PKI by assuring the domain administrator that the MASA service will only respect claims from authorized Registration Authorities of the domain. (The requirement for the Registrar to include the Domain CA certificate in the signature structure was stated above).

Registrar revocation consistency: The MASA SHOULD check for revocation of the Registrar certificate. The maximum lifetime of the voucher issued SHOULD NOT exceed the lifetime of the Registrar's revocation validation (for example if the Registrar revocation status is indicated in a CRL that is valid for two weeks then that is an appropriate lifetime for the voucher). Because the Registar certificate authority is unknown to the MASA in advance this is only an extended consistency check and is not required. The maximum lifetime of the voucher issued SHOULD NOT exceed the lifetime of the Registrar's revocation validation (for example if the Registrar revocation status is indicated in a CRL that is valid for two weeks then that is an appropriate lifetime for the voucher).

Pledge proximity assertion: The MASA server MAY verify that the Registrar signed voucher includes the 'prior-signed-voucher' field populated with a Pledge signed voucher that includes a 'proximity-registrar-cert' that is consistent with the certificate the Registrar used to sign the voucher request. The MASA server is aware of which Pledge's support signing of their voucher requests and can use this information to confirm proximity of the Pledge with the Registrar.

The Registrar certificate chain root certificate is extracted from the signature method and used to populate the "pinned-domain-cert" of the Voucher being issued. The domain ID (e.g. hash of the public key of the domain) is extracted from the root certificate and is used to update the audit log.

5.5. Voucher Response

The voucher response to requests from the Pledge and requests from a Registrar are in the same format. A Registrar either caches prior MASA responses or dynamically requests a new Voucher based on local policy.

If the join operation is successful, the server response MUST contain an HTTP 200 response code. The server MUST answer with a suitable 4xx or 5xx HTTP [RFC2616] error code when a problem occurs. The response data from the MASA server MUST be a plaintext human-readable (ASCII, english) error message containing explanatory information describing why the request was rejected.

A 403 (Forbidden) response is appropriate if the voucher request is not signed correctly, stale, or if the pledge has another outstanding voucher which can not be overridden.

A 404 (Not Found) response is appropriate when the request is for a device which is not known to the MASA.

A 406 (Not Acceptable) response is appropriate if a voucher of the desired type, or using the desired algorithms (as indicated by the Accept: headers, and algorithms used in the signature) can not be issued, such as because the MASA knows the pledge can not process that type.

A 415 (Unsupported Media Type) response is approriate for a request that has a voucher encoding that is not understood.

The response media type is:

application/pkcs7-mime; smime-type=voucher The response is a "YANG-defined JSON document that has been signed using a PKCS#7 structure" as described in [I-D.ietf-anima-voucher] using the JSON encoded described in [RFC7951]. The MASA MUST sign the request.

The syntactic details of vouchers are described in detail in [I-D.ietf-anima-voucher]. For example, the voucher consists of:

```
{
  "ietf-voucher:voucher": {
    "nonce": "62a2e7693d82fcda2624de58fb6722e5",
    "assertion": "logging"
    "pinned-domain-cert": "base64encodedvalue=="
    "serial-number": "JADA123456789"
  }
}
```

The Pledge verifies the signed voucher using the manufacturer installed trust anchor associated with the vendor's selected Manufacturer Authorized Signing Authority.

The 'pinned-domain-cert' element of the voucher contains the domain CA's public key. The Pledge MUST use the 'pinned-domain-cert' trust anchor to immediately complete authentication of the provisional TLS connection.

The Pledge MUST be prepared to parse and fail gracefully from a Voucher response that does not contain a 'pinned-domain-cert' field. The Pledge MUST be prepared to ignore additional fields it does not recognize.

5.5.1. Completing authentication of Provisional TLS connection

If a Registrar's credentials can not be verified using the pinned-domain-cert trust anchor from the voucher then the TLS connection is immediately discarded and the Pledge abandons attempts to bootstrap with this discovered registrar. The pledge SHOULD send voucher status telemetry (described below) before closing the TLS connection. The pledge MUST attempt to enroll using any other proxies it has found. It SHOULD return to the same proxy again after attempting with other proxies. Attempts should be attempted in the exponential backoff described earlier. Attempts SHOULD be repeated as failure may be the result of a temporary inconsistently (an inconsistently rolled Registrar key, or some other mis-configuration). The inconsistently could also be the result an active MITM attack on the EST connection.

The Registrar MUST use a certificate that chains to the pinned-domain-cert as its TLS server certificate.

The Pledge's PKIX path validation of a Registrar certificate's validity period information is as described in Section 2.5. Once the PKIX path validation is successful the TLS connection is no longer provisional.

The pinned-domain-cert is installed as an Explicit Trust Anchor for future operations. It can therefore can be used to authenticate any dynamically discovered EST server that contain the id-kp-cmcRA extended key usage extension as detailed in EST RFC7030 section 3.6.1; but to reduce system complexity the Pledge SHOULD avoid additional discovery operations. Instead the Pledge SHOULD communicate directly with the Registrar as the EST server. The ' pinned-domain-cert' is not a complete distribution of the EST section 4.1.3 CA Certificate Response which is an additional justification for the recommendation to proceed with EST key management operations. Once a full CA Certificate Response is obtained it is more authoritative for the domain than the limited 'pinned-domain-cert' response.'

5.6. Voucher Status Telemetry

The domain is expected to provide indications to the system administrators concerning device lifecycle status. To facilitate this it needs telemetry information concerning the device's status.

To indicate Pledge status regarding the Voucher, the pledge MUST post a status message.

The posted data media type: application/json

The client HTTP POSTs the following to the server at the EST well known URI /voucher_status. The Status field indicates if the Voucher was acceptable. If it was not acceptable the Reason string indicates why. In the failure case this message is being sent to an unauthenticated, potentially malicious Registrar and therefore the Reason string SHOULD NOT provide information beneficial to an attacker. The operational benefit of this telemetry information is balanced against the operational costs of not recording that an Voucher was ignored by a client the registar expected to continue joining the domain.

```
{
  "version":"1",
  "Status":FALSE /* TRUE=Success, FALSE=Fail"
  "Reason":"Informative human readable message"
  "reason-context": { additional JSON }
}
```

The server SHOULD respond with an HTTP 200 but MAY simply fail with an HTTP 404 error. The client ignores any response. Within the server logs the server SHOULD capture this telemetry information.

The reason-context attribute is an arbitrary JSON object (literal value or hash of values) which provides additional information specific to this pledge. The contents of this field are not subject to standardization."

Additional standard responses MAY be added via Specification Required.

5.7. MASA authorization log Request

After receiving the voucher status telemetry Section 5.6, the Registrar SHOULD request the MASA authorization log from the MASA service using this EST extension. If a device had previously registered with another domain, a Registrar of that domain would show in the log.

This is done with an HTTP GET using the operation path value of "/requestauditlog".

The registrar MUST HTTP POSTs the same Voucher Request as when requesting a Voucher. It is posted to the /requestauditlog URI instead. The "idevid-issuer" and "serial-number" informs the MASA server which log is requested so the appropriate log can be prepared for the response. Using the same media type and message minimizes cryptographic and message operations although it results in additional network traffic. The relying MASA server implementation MAY leverage internal state to associate this request with the original, and by now already validated, voucher request so as to avoid an extra crypto validation.

The request media type is:

application/pkcs7-mime; smime-type=voucher-request The request is a "YANG-defined JSON document that has been signed using a PKCS#7 structure" as described in Section 3 using the JSON encoded described in [RFC7951]. The Registrar MUST sign the request. The

entire Registrar certificate chain, up to and including the Domain CA, MUST be included in the PKCS#7 structure.

5.7.1. MASA authorization log Response

A log data file is returned consisting of all log entries. For example:

```
{
  "version":"1",
  "events":[
    {
     "date":"<date/time of the entry>",
     "domainID":"<domainID as extracted from the domain CA certificate
                 within the CMS of the audit voucher request>",
     "nonce":"<any nonce if supplied (or the exact string 'NULL')>"
    },
    {
     "date":"<date/time of the entry>",
     "domainID":"<domainID as extracted from the domain CA certificate
                 within the CMS of the audit voucher request>",
     "nonce":"<any nonce if supplied (or the exact string 'NULL')>"
    }
  ]
}
```

Distribution of a large log is less than ideal. This structure can be optimized as follows: All nonce-less entries for the same domainID MAY be condensed into the single most recent nonceless entry.

A Registrar SHOULD use this log information to make an informed decision regarding the continued bootstrapping of the Pledge. For example if the log includes unexpected domainIDs this is indicative of problematic imprints by the Pledge. If the log includes nonce-less entries this is indicative of the permanent ability for the indicated domain to trigger a reset of the device and take over management of it. Equipment that is purchased pre-owned can be expected to have an extensive history. A Registrar MAY request logs at future times. A Registrar MAY be configured to ignore the history of the device but it is RECOMMENDED that this only be configured if hardware assisted NEA [RFC5209] is supported.

Log entries containing the Domain's ID can be compared against local history logs in search of discrepancies.

This document specifies a simple log format as provided by the MASA service to the registar. This format could be improved by distributed consensus technologies that integrate vouchers with a

technologies such as block-chain or hash trees or the like. Doing so is out of the scope of this document but are anticipated improvements for future work. As such, the Registrar client SHOULD anticipate new kinds of responses, and SHOULD provide operator controls to indicate how to process unknown responses.

5.8. EST Integration for PKI bootstrapping

The Pledge SHOULD follow the BRSKI operations with EST enrollment operations including "CA Certificates Request", "CSR Attributes" and "Client Certificate Request" or "Server-Side Key Generation" etc. This is a relatively seamless integration since BRSKI REST calls provide an automated alternative to the manual bootstrapping method described in [RFC7030]. As noted above, use of HTTP 1.1 persistent connections simplifies the Pledge state machine.

The Pledge is also RECOMMENDED to implement the following EST automation extensions. They supplement the RFC7030 EST to better support automated devices that do not have an end user.

Although EST allows clients to obtain multiple certificates by sending multiple CSR requests BRSKI mandates use of the CSR Attributes request and mandates that the Registrar validate the CSR against the expected attributes. This implies that client requests will "look the same" and therefore result in a single logical certificate being issued even if the client were to make multiple requests. Registrars MAY contain more complex logic but doing so is out-of-scope of this specification. BRSKI does not signal any enhancement or restriction to this capability. Pledges that require multiple certificates could establish direct EST connections to the Registrar.

5.8.1. EST Distribution of CA Certificates

The Pledge MUST request the full EST Distribution of CA Certificates message. See RFC7030, section 4.1.

This ensures that the Pledge has the complete set of current CA certificates beyond the pinned-domain-cert (see Section 5.5.1 for a discussion of the limitations inherent in having a single certificate instead of a full CA Certificates response). Although these limitations are acceptable during initial bootstrapping they are not appropriate for ongoing PKIX end entity certificate validation.

5.8.2. EST CSR Attributes

Automated bootstrapping occurs without local administrative configuration of the Pledge. In some deployments its plausible that the Pledge generates a certificate request containing only identity information known to the Pledge (essentially the X.509 IDevID information) and ultimately receives a certificate containing domain specific identity information. Conceptually the CA has complete control over all fields issued in the end entity certificate. Realistically this is operationally difficult with the current status of PKI certificate authority deployments where the CSR is submitted to the CA via a number of non-standard protocols. Even with all standardized protocols used, it could operationally be problematic to expect that service specific certificate fields can be created by a CA that is likely operated by a group that has no insight into different network services/protocols used. For example, the CA could even be outsourced.

To alleviate these operational difficulties, the Pledge MUST request the EST "CSR Attributes" from the EST server and the EST server needs to be able to reply with the attributes necessary for use of the certificate in its intended protocols/services. This approach allows for minimal CA integrations and instead the local infrastructure (EST server) informs the Pledge of the proper fields to include in the generated CSR. This approach is beneficial to automated boostrapping in the widest number of environments.

If the hardwareModuleName in the X.509 IDevID is populated then it SHOULD by default be propagated to the LDevID along with the hwSerialNum. The EST server SHOULD support local policy concerning this functionality.

In networks using the BRSKI enrolled certificate to authenticate the ACP (Autonomic Control Plane), the EST attributes MUST include the "ACP information" field. See [I-D.ietf-anima-autonomic-control-plane] for more details.

The Registar MUST also confirm the resulting CSR is formatted as indicated before forwarding the request to a CA. If the Registar is communicating with the CA using a protocol like full CMC which provides mechanisms to override the CSR attributes, then these mechanisms MAY be used even if the client ignores CSR Attribute guidance.

5.8.3. EST Client Certificate Request

The Pledge MUST request a new client certificate. See RFC7030, section 4.2.

5.8.4. Enrollment Status Telemetry

For automated bootstrapping of devices the adminstrative elements providing bootstrapping also provide indications to the system administrators concerning device lifecycle status. This might include information concerning attempted bootstrapping messages seen by the client, MASA provides logs and status of credential enrollment. The EST protocol assumes an end user and therefore does not include a final success indication back to the server. This is insufficient for automated use cases.

To indicate successful enrollment the client SHOULD re-negotiate the EST TLS session using the newly obtained credentials. This occurs by the client initiating a new TLS ClientHello message on the existing TLS connection. The client MAY simply close the old TLS session and start a new one. The server MUST support either model.

In the case of a FAIL the Reason string indicates why the most recent enrollment failed. The SubjectKeyIdentifier field MUST be included if the enrollment attempt was for a keypair that is locally known to the client. If EST /serverkeygen was used and failed then the field is omitted from the status telemetry.

In the case of a SUCCESS the Reason string is omitted. The SubjectKeyIdentifier is included so that the server can record the successful certificate distribution.

Status media type: application/json

The client HTTP POSTs the following to the server at the new EST well known URI /enrollstatus.

```
{
  "version":"1",
  "Status":TRUE /* TRUE=Success, FALSE=Fail"
  "Reason":"Informative human readable message"
  "reason-context": "Additional information"
}
```

The server SHOULD respond with an HTTP 200 but MAY simply fail with an HTTP 404 error.

Within the server logs the server MUST capture if this message was received over an TLS session with a matching client certificate. This allows for clients that wish to minimize their crypto operations to simply POST this response without renegotiating the TLS session - at the cost of the server not being able to accurately verify that enrollment was truly successful.

5.8.5. EST over CoAP

This document describes extensions to EST for the purposes of bootstrapping of remote key infrastructures. Bootstrapping is relevant for CoAP enrollment discussions as well. The defintion of EST and BRSKI over CoAP is not discussed within this document beyond ensuring proxy support for CoAP operations. Instead it is anticipated that a definition of CoAP mappings will occur in subsequent documents such as [I-D.vanderstok-ace-coap-est] and that CoAP mappings for BRSKI will be discussed either there or in future work.

6. Reduced security operational modes

A common requirement of bootstrapping is to support less secure operational modes for support specific use cases. The following sections detail specific ways that the Pledge, Registrar and MASA can be configured to run in a less secure mode for the indicated reasons.

6.1. Trust Model

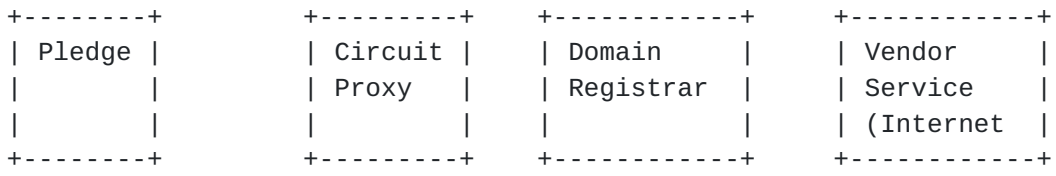


Figure 10

Pledge: The Pledge could be compromised and providing an attack vector for malware. The entity is trusted to only imprint using secure methods described in this document. Additional endpoint assessment techniques are RECOMMENDED but are out-of-scope of this document.

Proxy: Provides proxy functionalities but is not involved in security considerations.

Registrar: When interacting with a MASA server a Registrar makes all decisions. When Ownership Vouchers are involved a Registrar is

only a conduit and all security decisions are made on the vendor service.

Vendor Service, MASA: This form of vendor service is trusted to accurately log all claim attempts and to provide authoritative log information to Registrars. The MASA does not know which devices are associated with which domains. These claims could be strengthened by using cryptographic log techniques to provide append only, cryptographic assured, publicly auditable logs. Current text provides only for a trusted vendor.

Vendor Service, Ownership Validation: This form of vendor service is trusted to accurately know which device is owned by which domain.

6.2. Pledge security reductions

The Pledge can choose to accept vouchers using less secure methods. These methods enable offline and emergency (touch based) deployment use cases:

1. The Pledge MUST accept nonceless vouchers. This allows for offline use cases. Logging and validity periods address the inherent security considerations of supporting these use cases.

2. The Pledge MAY support "trust on first use" for physical interfaces such as a local console port or physical user interface but MUST NOT support "trust on first use" on network interfaces. This is because "trust on first use" permanently degrades the security for all use cases.

3. The Pledge MAY have an operational mode where it skips Voucher validation one time. For example if a physical button is depressed during the bootstrapping operation. This can be useful if the vendor service is unavailable. This behavior SHOULD be available via local configuration or physical presence methods to ensure new entities can always be deployed even when autonomic methods fail. This allows for unsecured imprint.

It is RECOMMENDED that "trust on first use" or skipping voucher validation only be available if hardware assisted Network Endpoint Assessment [RFC5209] is supported. This recommendation ensures that domain network monitoring can detect innappropriate use of offline or emergency deployment procedures.

6.3. Registrar security reductions

A Registrar can choose to accept devices using less secure methods. These methods are acceptable when low security models are needed, as the security decisions are being made by the local administrator, but they MUST NOT be the default behavior:

1. A registrar MAY choose to accept all devices, or all devices of a particular type, at the administrator's discretion. This could occur when informing all Registrars of unique identifiers of new entities might be operationally difficult.

2. A registrar MAY choose to accept devices that claim a unique identity without the benefit of authenticating that claimed identity. This could occur when the Pledge does not include an X.509 IDevID factory installed credential. New Entities without an X.509 IDevID credential MAY form the Section 5.2 request using the Section 5.4 format to ensure the Pledge's serial number information is provided to the Registar (this includes the IDevID AuthorityKeyIdentifier value which would be statically configured on the Pledge). The Pledge MAY refuse to provide a TLS client certificate (as one is not available). The Pledge SHOULD support HTTP-based or certificate-less TLS authentication as described in EST RFC7030 section 3.3.2. A Registrar MUST NOT accept unauthenticated New Entities unless it has been configured to do so by an administrator that has verified that only expected new entities can communicate with a Registrar (presumably via a physically secured perimeter).

3. A Registrar MAY request nonce-less Vouchers from the MASA service (by not including a nonce in the request). These Vouchers can then be transmitted to the Registrar and stored until they are needed during bootstrapping operations. This is for use cases where target network is protected by an air gap and therefore can not contact the MASA service during Pledge deployment.

4. A registrar MAY ignore unrecognized nonce-less log entries. This could occur when used equipment is purchased with a valid history being deployed in air gap networks that required permanent Vouchers.

6.4. MASA security reductions

Lower security modes chosen by the MASA service effect all device deployments unless bound to the specific device identities. In which case these modes can be provided as additional features for specific customers. The MASA service can choose to run in less secure modes by:

1. Not enforcing that a nonce is in the Voucher. This results in distribution of Voucher that never expires and in effect makes the Domain an always trusted entity to the Pledge during any subsequent bootstrapping attempts. That this occurred is captured in the log information so that the Registrar can make appropriate security decisions when a Pledge joins the Domain. This is useful to support use cases where Registrars might not be online during actual device deployment. Because this results in long lived Voucher and does not require the proof that the device is online this is only accepted when the Registrar is authenticated by the MASA server and authorized to provide this functionality. The MASA server is RECOMMENDED to use this functionality only in concert with an enhanced level of ownership tracking (out-of-scope). If the Pledge device is known to have a real-time-clock that is set from the factory use of a voucher validity period is RECOMMENDED.

2. Not verifying ownership before responding with an Voucher. This is expected to be a common operational model because doing so relieves the vendor providing MASA services from having to track ownership during shipping and supply chain and allows for a very low overhead MASA service. A Registrar uses the audit log information as a defense in depth strategy to ensure that this does not occur unexpectedly (for example when purchasing new equipment the Registrar would throw an error if any audit log information is reported). The MASA should verify the 'prior-signed-voucher' information for Pledge's that support that functionality. This provides a proof-of-proximity check that reduces the need for ownership verification.

7. IANA Considerations

This document requests the following Parameter Values for the "smime-type" Parameters:

- voucher-request

- voucher

7.1. PKIX Registry

IANA is requested to register the following:

This document requests a number for id-mod-MASAURLExtn2016(TBD) from the pkix(7) id-mod(0) Registry. [[EDNOTE: fix names]]

This document requests a number from the id-pe registry for id-pe-masa-url. XXX

7.2. MIME

Type name:

Subtype name:

Required parameters:

Optional parameters:

Encoding considerations:

Security considerations:

Interoperability considerations:

Published specification:

Applications that use this media type:

Fragment identifier considerations:

Additional information:

Deprecated alias names for this type:

Magic number(s):

File extension(s):

Macintosh file type code(s):

Person and email address to contact for further information:

Intended usage: LIMITED USED

Restrictions on usage:

Author:

Change controller:

Provisional registration? (standards tree only):

7.3. Voucher Status Telemetry

IANA is requested to create a registry entitled: _Voucher Status Telemetry Attributes_. New items can be added using the Specification Required. The following items are to be in the initial registration, with this document as the reference:

- version
- Status
- Reason
- reason-context

8. Security Considerations

There are uses cases where the MASA could be unavailable or uncooperative to the Registrar. They include planned and unplanned network partitions, changes to MASA policy, or other instances where MASA policy rejects a claim. These introduce an operational risk to the Registrar owner that MASA/vendor behavior might limit the ability to re-boostrap a Pledge device. For example this might be an issue during disaster recovery. This risk can be mitigated by Registrars that request and maintain long term copies of "nonceless" Vouchers. In that way they are guaranteed to be able to repeat bootstrapping for their devices.

The issuance of nonceless vouchers themselves create a security concern. If the Registrar of a previous domain can intercept protocol communications then it can use a previously issued nonceless voucher to establish management control of a pledge device even after having sold it. This risk is mitigated by recording the issuance of such vouchers in the MASA audit log that is verified by the subsequent Registrar. This reduces the resale value of the equipment because future owners will detect the lowered security inherent in the existence of a nonceless voucher that would be trusted by their Pledge. This reflects a balance between partition resistant recovery and security of future bootstrapping. Registrars take the Pledge's audit history into account when applying policy to new devices.

The MASA server is exposed to DoS attacks wherein attackers claim an unbounded number of devices. Ensuring a Registrar is representative of a valid vendor customer, even without validating ownership of specific Pledge devices, helps to mitigate this. Pledge signatures on the initial voucher request, as forwarded by the Registrar in the prior-signed-voucher field, significantly reduce this risk by ensuring the MASA can confirm proximity between the Pledge and the

Registrar making the request. This mechanism is optional to allow for constrained devices.

It is possible for an attacker to request a voucher from the MASA service directly after the real Registrar obtains an audit log. If the attacker could also force the bootstrapping protocol to reset there is a theoretical opportunity for the attacker to use their voucher to take control of the Pledge but then proceed to enroll with the target domain. Possible prevention mechanisms include:

- Per device rate limits on the MASA service ensure such timing attacks are difficult.

- The Registrar can repeat the request for audit log information at some time after bootstrapping is complete.

To facilitate logging and administrative oversight the Pledge reports on Voucher parsing status to the Registrar. In the case of a failure this information is informative to a potentially malicious Registar but this is RECOMMENDED anyway because of the operational benefits of an informed administrator in cases where the failure is indicative of a problem.

To facilitate truely limited clients EST RFC7030 section 3.3.2 requirements that the client MUST support a client authentication model have been reduced in Section 6 to a statement that the Registrar "MAY" choose to accept devices that fail cryptographic authentication. This reflects current (poor) practices in shipping devices without a cryptographic identity that are NOT RECOMMENDED.

During the provisional period of the connection all HTTP header and content data MUST treated as untrusted data. HTTP libraries are regularly exposed to non-secured HTTP traffic: mature libraries should not have any problems.

Pledge's might chose to engage in protocol operations with multiple discovered Registrars in parallel. As noted above they will only do so with distinct nonce values, but the end result could be multple voucher's issued from the MASA if all registrars attempt to claim the device. This is not a failure and the Pledge choses whichever voucher to accept based on internal logic. The Registrar's verifying log information will see multiple entries and take this into account for their analytics purposes.

8.1. Freshness in Voucher Requests

A concern has been raised that the voucher request produced by the Pledge should contain some content (a nonce) from the Registrar and/or MASA in order for those actors to verify that the voucher request is fresh.

There are a number of operational problems with getting a nonce from the MASA to the pledge. It is somewhat easier to collect a random value from the Registrar, but as the Registrar is not yet vouched for, such a Registrar nonce has little value. There are privacy and logistical challenges to addressing these operational issues, so if such a thing were to be considered, it would have to provide some clear value. This section examines the impacts of not having a fresh voucher request from the pledge.

Because the Registrar authenticates the Pledge a full Man-in-the-Middle attack is not possible, despite the provisional TLS authentication by the Pledge (see Section 5). Instead we examine the case of a fake Registrar (Rm) that communicates with the Pledge in parallel or in close time proximity with the intended Registrar. (This scenario is intentionally supported as described in Section 4.1).

The fake Registrar (Rm) can obtain a voucher signed by the MASA either directly or through arbitrary intermediaries. Assuming that the MASA accepts the voucher request (either because Rm is collaborating with a legitimate Registrar according to supply chain information, or because the MASA is in audit-log only mode), then a voucher linking the pledge to the Registrar Rm is issued.

Such a voucher, when passed back to the Pledge, would link the pledge to Registrar Rm, and would permit the Pledge to end the provisional state. It now trusts Rm and, if it has any security vulnerabilities leveragable by an Rm with full administrative control, can be assumed to be a threat against the intended Registrar.

This flow is mitigated by the intended Registar verifying the audit logs available from the MASA as described in Section 5.7. Rm might chose to wait until after the intended Registrar completes the authorization process before submitting the now-stale voucher request. The Rm would need to remove the Pledge's nonce.

In order to successfully use the resulting "stale voucher" Rm would have to attack the Pledge and return it to a bootstrapping enabled state. This would require wiping the Pledge of current configuration and triggering a re-bootstrapping of the Pledge. This is no more likely than simply taking control of the Pledge directly but if this

is a consideration the target network is RECOMMENDED to take the following steps:

- Ongoing network monitoring for unexpected bootstrapping attempts by Pledges.

- Retreival and examination of MASA log information upon the occurance of any such unexpected events. Rm will be listed in the logs.

9. Acknowledgements

We would like to thank the various reviewers for their input, in particular Brian Carpenter, Toerless Eckert, Fuyu Eleven, Eliot Lear, Sergey Kasatkin, Markus Stenberg, and Peter van der Stok

10. References

10.1. Normative References

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-10 (work in progress), September 2017.

[I-D.ietf-anima-voucher]
Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "Voucher Profile for Bootstrapping Protocols", draft-ietf-anima-voucher-05 (work in progress), August 2017.

[I-D.vanderstok-ace-coap-est]
Kumar, S., Stok, P., Kampanakis, P., Furuhed, M., and S. Raza, "EST over secure CoAP (EST-coaps)", draft-vanderstok-ace-coap-est-02 (work in progress), June 2017.

[IDevID] IEEE Standard, "IEEE 802.1AR Secure Device Identifier", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[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>.

[RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, DOI 10.17487/RFC3542, May 2003, <https://www.rfc-editor.org/info/rfc3542>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <https://www.rfc-editor.org/info/rfc3927>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <https://www.rfc-editor.org/info/rfc4941>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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>.

[RFC5386] Williams, N. and M. Richardson, "Better-Than-Nothing Security: An Unauthenticated Mode of IPsec", RFC 5386, DOI 10.17487/RFC5386, November 2008, <https://www.rfc-editor.org/info/rfc5386>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5660] Williams, N., "IPsec Channels: Connection Latching", RFC 5660, DOI 10.17487/RFC5660, October 2009, <https://www.rfc-editor.org/info/rfc5660>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[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>.

[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>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

[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>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

10.2. Informative References

[I-D.behringer-homenet-trust-bootstrap]
Behringer, M., Pritikin, M., and S. Bjarnason, "Bootstrapping Trust on a Homenet", draft-behringer-homenet-trust-bootstrap-02 (work in progress), February 2014.

[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-netconf-zerotouch]
Watsen, K., Abrahamsson, M., and I. Farrer, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-17 (work in progress), September 2017.

[I-D.ietf-opsawg-mud]
Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", draft-ietf-opsawg-mud-12 (work in progress), October 2017.

[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-01 (work in progress), July 2016.

[imprinting]
Wikipedia, "Wikipedia article: Imprinting", July 2015, <https://en.wikipedia.org/wiki/Imprinting_(psychology)>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <https://www.rfc-editor.org/info/rfc7217>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/info/rfc7435>.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <https://www.rfc-editor.org/info/rfc7575>.

[Stajano99theresurrecting]
Stajano, F. and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks", 1999, <https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf>.

Appendix A. IPv4 operations

A.1. IPv4 Link Local addresses

Instead of an IPv6 link-local address, an IPv4 address may be generated using [RFC3927] Dynamic Configuration of IPv4 Link-Local Addresses.

In the case that an IPv4 Local-Local address is formed, then the bootstrap process would continue as in the IPv6 case by looking for a (circuit) proxy.

A.2. Use of DHCPv4

The Plege MAY obtain an IP address via DHCP [RFC2131]. The DHCP provided parameters for the Domain Name System can be used to perform DNS operations if all local discovery attempts fail.

Appendix B. mDNS / DNSSD proxy discovery options

The Pledge MAY perform DNS-based Service Discovery [RFC6763] over Multicast DNS [RFC6762] searching for the service "_bootstrapks._tcp.local.".

To prevent unaccceptable levels of network traffic the congestion avoidance mechanisms specified in [RFC6762] section 7 MUST be followed. The Pledge SHOULD listen for an unsolicited broadcast response as described in [RFC6762]. This allows devices to avoid announcing their presence via mDNS broadcasts and instead silently join a network by watching for periodic unsolicited broadcast responses.

Performs DNS-based Service Discovery [RFC6763] over normal DNS operations. The service searched for is "_bootstrapks._tcp.example.com". In this case the domain "example.com" is discovered as described in [RFC6763] section 11. This method is only available if the host has received a useable IPv4 address via DHCPv4 as suggested in Appendix A.

If no local bootstrapks service is located using the GRASP mechanisms, or the above mentioned DNS-based Service Discovery methods the Pledge MAY contact a well known vendor provided bootstrapping server by performing a DNS lookup using a well known URI such as "bootstrapks.vendor-example.com". The details of the URI are vendor specific. Vendors that leverage this method on the Pledge are responsible for providing the bootstrapks service.

The current DNS services returned during each query is maintained until bootstrapping is completed. If bootstrapping fails and the Pledge returns to the Discovery state it picks up where it left off and continues attempting bootstrapping. For example if the first Multicast DNS _bootstrapks._tcp.local response doesn't work then the second and third responses are tried. If these fail the Pledge moves on to normal DNS-based Service Discovery.

Appendix C. IPIP Join Proxy mechanism

The Circuit Proxy mechanism suffers from requiring a state on the Join Proxy for each connection that is relayed. The Circuit Proxy can be considered a kind of Algorithm Gateway [FIND-good-REF].

An alternative to proxying at the TCP layer is to selectively forward at the IP layer. This moves all per-connection to the Join Registrar. The IPIP tunnel statelessly forwards packets. This section provides some explanation of some of the details of the Registrar discovery procotol which are not important to Circuit Proxy, and some implementation advice.

The IPIP tunnel is described in [RFC2473]. Each such tunnel is considered a unidirectional construct, but two tunnels may be associated to form a bidirectional mechanism. An IPIP tunnel is setup as follows. The outer addresses are an ACP address of the Join Proxy, and the ACP address of the Join Registrar. The inner addresses seen in the tunnel are the link-local addresses of the network on which the join activity is occuring.

One way to look at this construct is to consider that the Registrar is extending attaching an interface to the network on which the Join Proxy is physically present. The Registrar then interacts as if it were present on that network using link-local (fe80::) addresses. The Join node is unaware that the traffic is being proxied through a tunnel, and does not need any special routing.

There are a number of considerations with this mechanism which require cause some minor amounts of complexity. Note that due to the tunnels, the Registrar sees multiple connections to a fe80::/10 network on not just physical interfaces, but on each of the virtual interfaces represending the tunnels.

C.1. Multiple Join networks on the Join Proxy side

The Join Proxy will in the general case be a routing device with multiple interfaces. Even a device as simple as a wifi access point may have wired, and multiple frequencies of wireless interfaces, potentially with multiple ESSIDs.

Each of these interfaces on the Join Proxy may be seperate L3 routing domains, and therefore will have a unique set of link-local addresses. An IPIP packet being returned by the Registrar needs to be forwarded to the correct interface, so the Join Proxy needs an additional key to distinguish which network the packet should be returned to.

The simplest way to get this additional key is to allocate an additional ACP address; one address for each network on which join traffic is occuring. The Join Proxy SHOULD do a GRASP M_NEG_SYN for each interface which they wish to relay traffic, as this allows the Registrar to do any static tunnel configuration that may be required.

C.2. Automatic configuration of tunnels on Registrar

The Join Proxy is expected to do a GRASP negotiation with the proxy for each Join Interface that it needs to relay traffic from. This is to permit Registrars to configure the appropriate virtual interfaces before join traffic arrives.

A Registrar serving a large number of interfaces may not wish to allocate resources to every interface at all times, but can instead dynamically allocate interfaces. It can do this by monitoring IPIP traffic that arrives on it's ACP interface, and when packets arrive from new Join Proxys, it can dynamically configure virtual interfaces.

A more sophisticated Registrar willing to modify the behaviour of it's TCP and UDP stack could note the IPIP traffic origination in the socket control block and make information available to the TCP layer (for HTTPS connections), or to the application (for CoAP connections) via a proprietary extension to the socket API.

C.3. Proxy Neighbor Discovery by Join Proxy

The Join Proxy MUST answer neighbor discovery messages for the address given by the Registrar as being it's link-local address. The Join Proxy must also advertise this address as the address to which to connect to when advertising it's existence.

This proxy neighbor discovery means that the pledge will create TCP and UDP connections to the correct Registrar address. This matters as the TCP and UDP pseudo-header checksum includes the destination address, and for the proxy to remain completely stateless, it must not be necessary for the checksum to be updated.

C.4. Use of connected sockets; or IP_PKTINFO for CoAP on Registrar

TCP connections on the registrar SHOULD properly capture the ifindex of the incoming connection into the socket structure. This is normal IPv6 socket API processing. The outgoing responses will go out on the same (virtual) interface by ifindex.

When using UDP sockets with CoAP, the application will have to pay attention to the incoming ifindex on the socket. Access to this

information is available using the IP_PKTINFO auxiliary extension which is a standard part of the IPv6 sockets API.

A registrar application could, after receipt of an initial CoAP message from the Pledge, create a connected UDP socket (including the ifindex information). The kernel would then take care of accurate demultiplexing upon receive, and subsequent transmission to the correct interface.

C.5. Use of socket extension rather than virtual interface

Some operating systems on which a Registrar need be implemented may find need for a virtual interface per Join Proxy to be problematic. There are other mechanism which can make be done.

If the IPIP decapsulator can mark the (SYN) packet inside the kernel with the address of the Join Proxy sending the traffic, then an interface per Join Proxy may not be needed. The outgoing path need just pay attention to this extra information and add an appropriate IPIP header on outgoing. A CoAP over UDP mechanism may need to expose this extra information to the application as the UDP sockets are often not connected, and the application will need to specify the outgoing path on each packet send.

Such an additional socket mechanism has not been standardized. Terminating L2TP connections over IPsec transport mode suffers from the same challenges.

Appendix D. MUD Extension

The following extension augments the MUD model to include a single node, as described in [I-D.ietf-opsawg-mud] section 3.6, using the following sample module that has the following tree structure:

```
module: ietf-mud-brski-masa
augment /ietf-mud:mud:
+--rw masa-server?   inet:uri
```

The model is defined as follows:

```
<CODE BEGINS>
module ietf-mud-brski-masa {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-brski-masa";
  prefix ietf-mud-brski-masa;
  import ietf-mud {
    prefix ietf-mud;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF ANIMA (Autonomic Networking Integrated Model and
    Approach) Working Group";
    contact
    "WG Web: http://tools.ietf.org/wg/anima/
    WG List: anima@ietf.org
    ";
  description
    "BRSKI extension to a MUD file to indicate the
    MASA URL.";

  revision 2017-10-09 {
    description
    "Initial revision.";
    reference
    "RFC XXXX: Manufacturer Usage Description
    Specification";
  }

  augment "/ietf-mud:mud" {
    description
    "BRSKI extension to a MUD file to indicate the
    MASA URL.";
    leaf masa-server {
      type inet:uri;
      description
      "This value is the URI of the MASA server";
    }
  }
}
<CODE ENDS>
```

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael C. Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

Steinthor Bjarnason
Arbor Networks

Email: sbjarnason@arbor.net

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net